credit_risk

June 30, 2025

Credit Risk Assessment

Project Goals:

```
Analyze borrower profiles and their financial behavior<br>
Create a classification model to identify reliable and high-risk clients.<br>
Build a dashboard to highlight segments that may be riskier, more profitable, or require diffe
```

Project Stages:

Data Preparation

```
Removing duplicates, handling missing values and outliers, analyzing variable distributions
```

Exploratory Analysis:

Financial Profile:

What is the average loan_percent_income across different income levels? Who most often takes out large loans — by age, experience, property type? How are income and interest rate (loan_int_rate) related? How does loan_percent_income vary depending on credit grade (loan_grade)?

Credit History and Risk:

How does credit history length (cred_hist_length) affect loan amount? What is the average credit history length across age groups? How is default history (cb_person_default_on_file) related to interest rate? Who has more defaults — by age, loan purpose, or employment status?

Dependence on Loan Purpose:

Which loan purposes are more often linked to large amounts? Which loan intents have a higher share of borrowers with default history?

For the Classification Model:

Encoding categorical variables and scaling all features Which features are most informative for predicting default? Which features are correlated? How does loan_grade affect default probability? Is there multicollinearity — especially among features like interest rate / amount / income?

Model Building

```
Choosing an algorithm and building the classifier<br>
Evaluating and improving the model<br>
Predicting client classes on new data<br>
```

Dashboard Creation

Creating visualizations:
What is the average loan amount by loan purpose?
How do interest rates differ by loan purpose?
How is borrower age distributed by loan purpose?
What is the average credit burden by loan purpose?
How does homeownership category vary by loan purpose?
How is credit score related to loan purpose?

[808]:
```
import numpy as np
import pandas as pd
import scipy as sp
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import randint
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split, cross_val_score, ␣
 ↪RandomizedSearchCV, GridSearchCV
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix, precision_recall_curve,\
roc_auc_score, roc_curve, fbeta_score, cohen_kappa_score, matthews_corrcoef
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
```

[810]:
```
df = pd.read_csv('credit_risk.csv')
```

[812]:
```
df
```

[812]:

	person_age	person_income	person_home_ownership	person_emp_length	\
0	26	200100	MORTGAGE	10.0	
1	31	64450	RENT	3.0	
2	22	22292	RENT	5.0	
3	33	109000	MORTGAGE	NaN	
4	24	100000	MORTGAGE	8.0	
…	…	…	…	…	
24430	39	38500	MORTGAGE	7.0	
24431	25	69000	RENT	5.0	
24432	26	148000	RENT	1.0	
24433	26	175000	MORTGAGE	0.0	
24434	27	65000	MORTGAGE	11.0	

	loan_intent	loan_grade	loan_amnt	loan_int_rate	loan_status	\
0	PERSONAL	B	30000	11.71	0	
1	VENTURE	C	12000	12.73	0	
2	DEBTCONSOLIDATION	D	3500	16.49	1	
3	MEDICAL	B	15000	11.49	0	
4	EDUCATION	A	16800	7.88	0	

```
…                  …        …        …              …           …
24430            MEDICAL          C       3500          13.98           0
24431    HOMEIMPROVEMENT          A       8500           6.92           1
24432  DEBTCONSOLIDATION          E      20000          17.99           1
24433           PERSONAL          C      15000            NaN           0
24434            VENTURE          A       8000           7.49           0

       loan_percent_income cb_person_default_on_file  \
0                     0.15                         N
1                     0.19                         N
2                     0.16                         N
3                     0.14                         N
4                     0.17                         N
…                      …                           …
24430                 0.09                         Y
24431                 0.12                         N
24432                 0.14                         N
24433                 0.09                         N
24434                 0.12                         N

       cb_person_cred_hist_length
0                               4
1                               8
2                               3
3                               5
4                               4
…                               …
24430                          17
24431                           4
24432                           3
24433                           3
24434                           8

[24435 rows x 12 columns]
```

Data preparing

[815]:
```
df.duplicated().sum()
```

[815]:
```
103
```

Removing duplicate rows

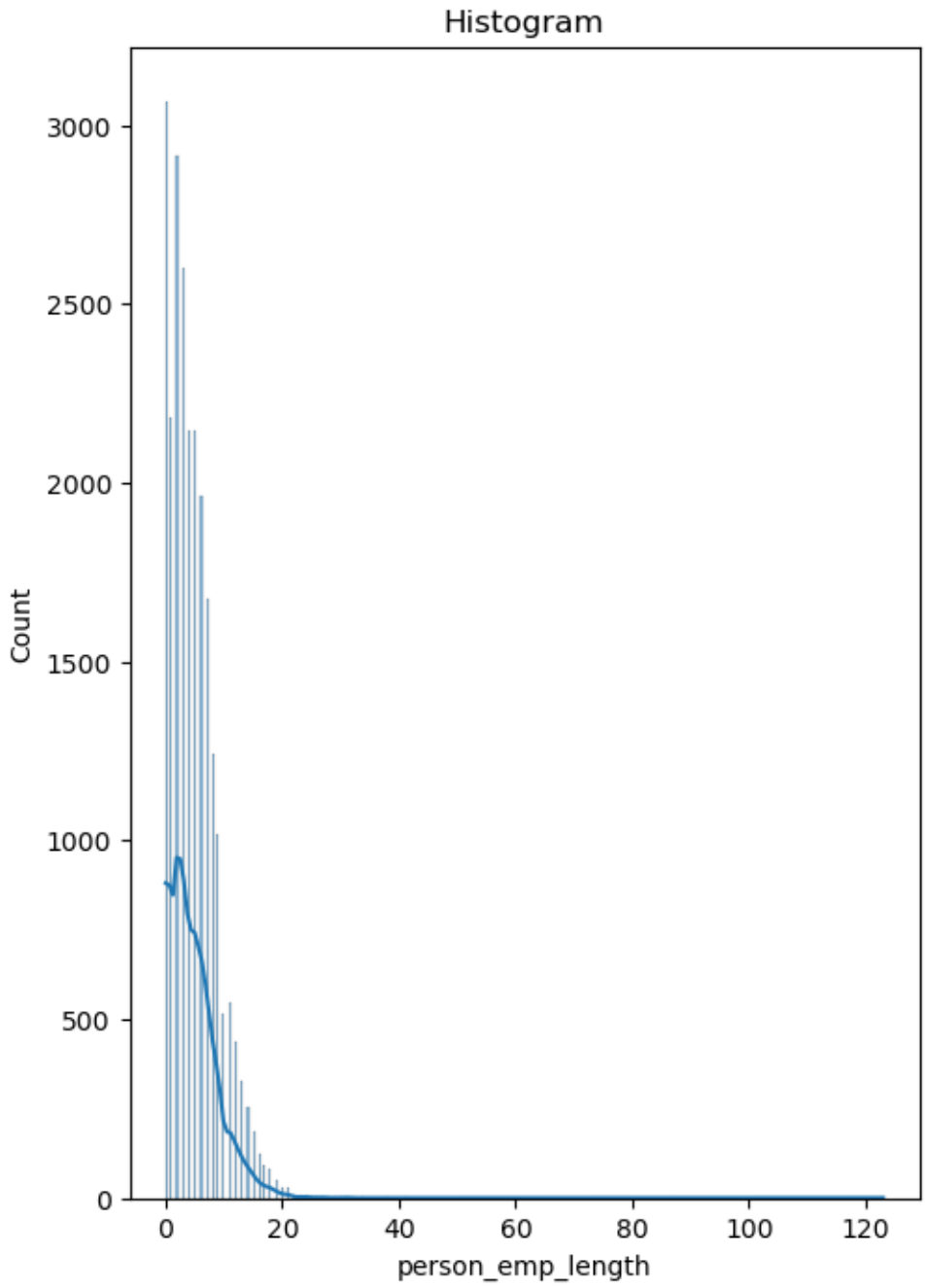
[818]:
```
df = df.drop_duplicates()
df.duplicated().sum()
```

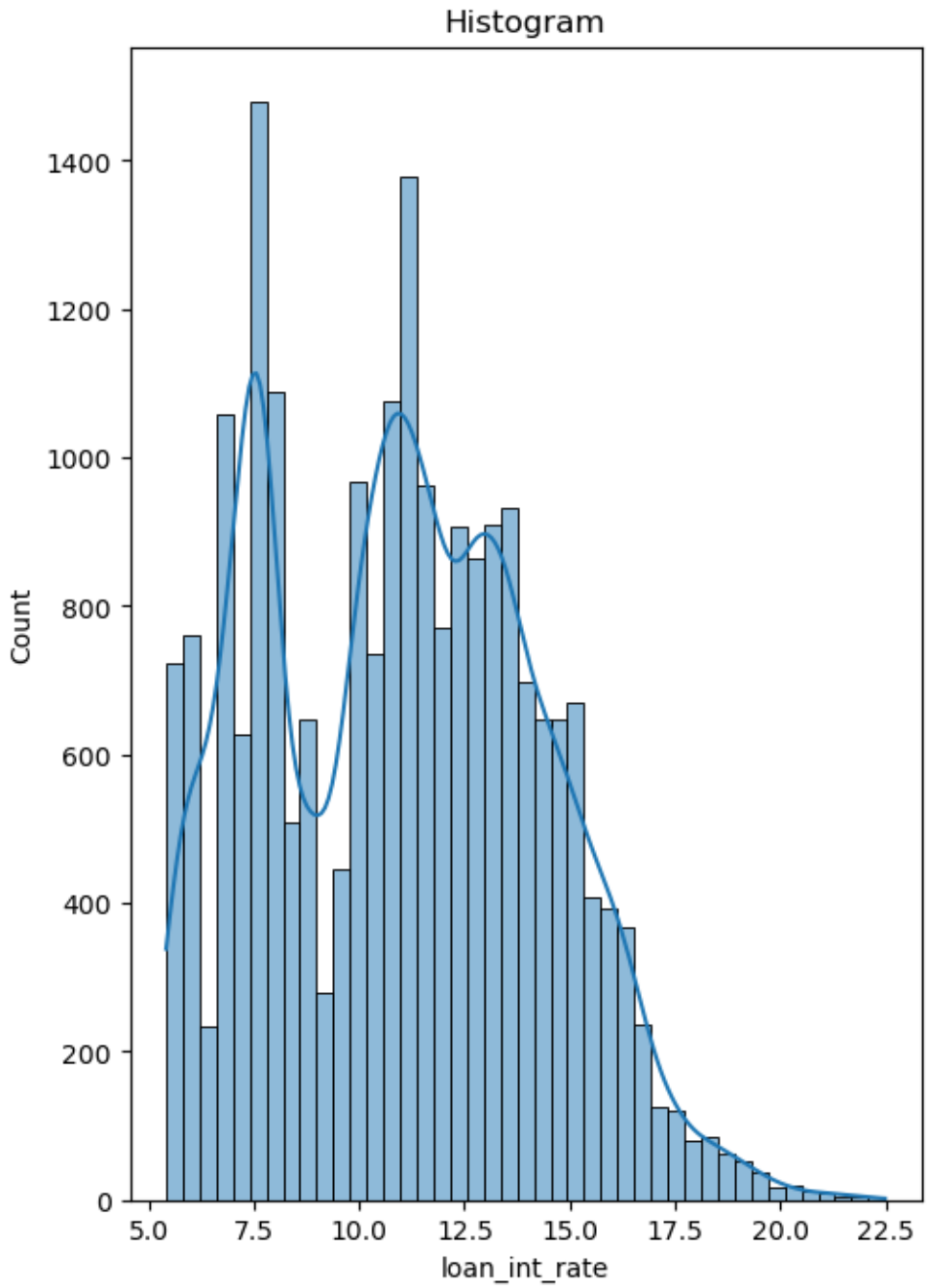
[818]:
```
0
```

[820]:
```
df.isnull().sum()
```

[820]:
```
person_age                      0
person_income                   0
person_home_ownership           0
person_emp_length             673
loan_intent                     0
loan_grade                      0
loan_amnt                       0
loan_int_rate                2295
loan_status                     0
loan_percent_income             0
cb_person_default_on_file       0
cb_person_cred_hist_length      0
dtype: int64
```

[822]:
```
columns = ['person_emp_length', 'loan_int_rate']

for column in columns:
    plt.figure(figsize=(12, 8))
    plt.subplot(1, 2, 1)
    sns.histplot(df[column].dropna(), kde=True)
    plt.title('Histogram')
    plt.show()
```


Histogram
Count
3000
2500
2000
1500
1000
500
0
0
20
40
60
80
100
120
person_emp_length



Let's replace the zero values with the median, since the distribution is not normal.

[824]:
```
for column in columns:
    df[column] = df[column].fillna(df[column].median())
```

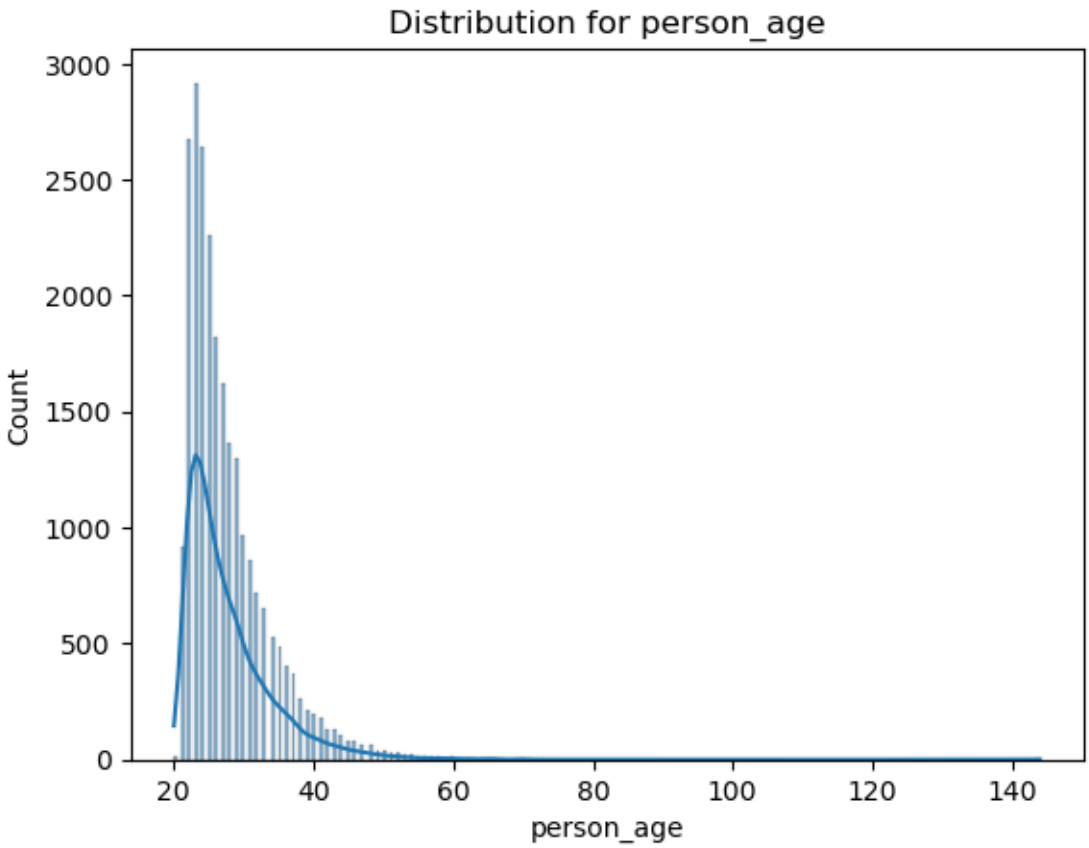
```
/var/folders/g_/dlksrxdd3pz88bqsmz91cx540000gn/T/ipykernel_28061/769256592.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

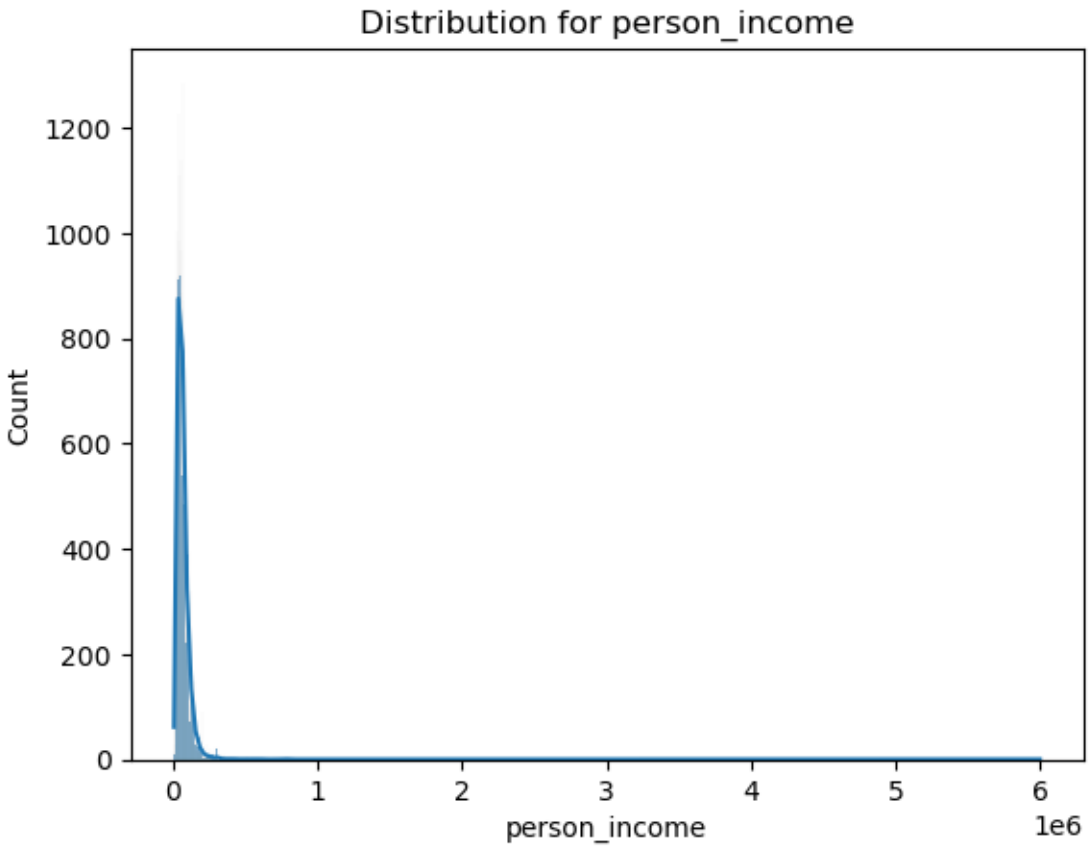
```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[column] = df[column].fillna(df[column].median())
```

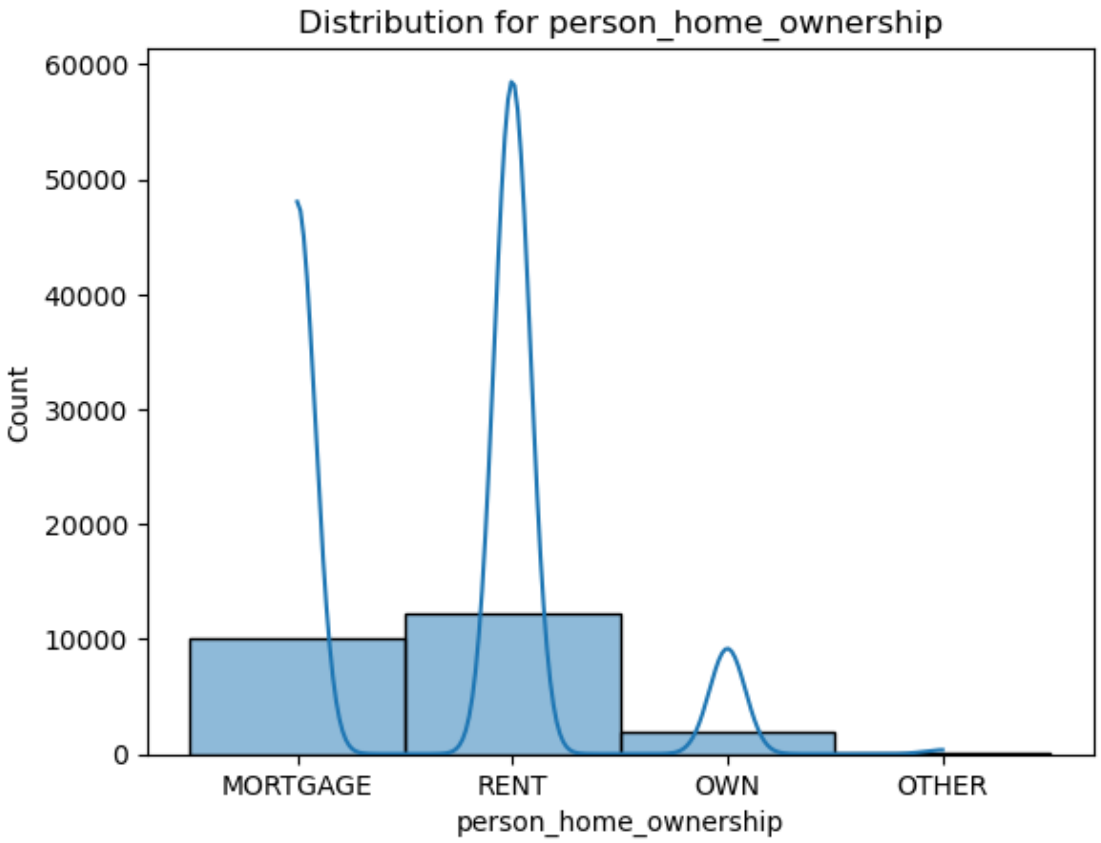
[825]:
```
df.isnull().sum()
```

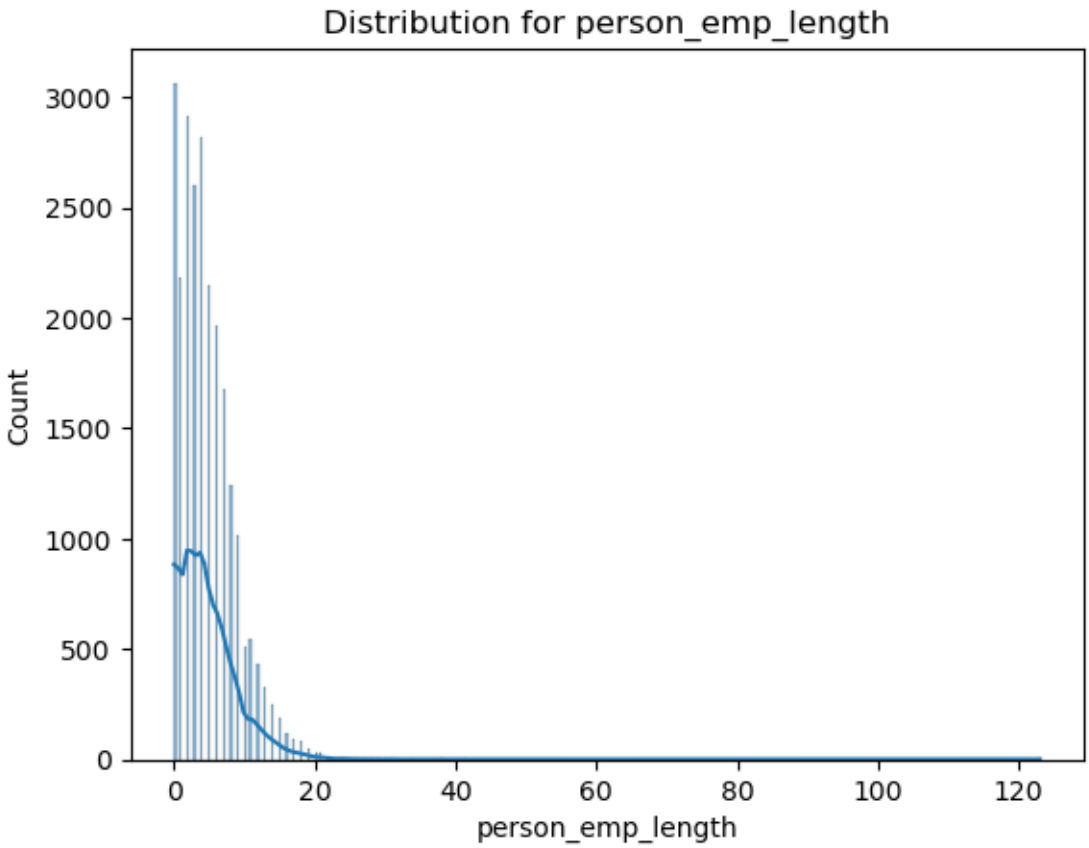
[825]:
```
person_age                  0
person_income               0
person_home_ownership       0
person_emp_length           0
loan_intent                 0
loan_grade                  0
loan_amnt                   0
loan_int_rate               0
loan_status                 0
loan_percent_income         0
cb_person_default_on_file   0
cb_person_cred_hist_length  0
dtype: int64
```

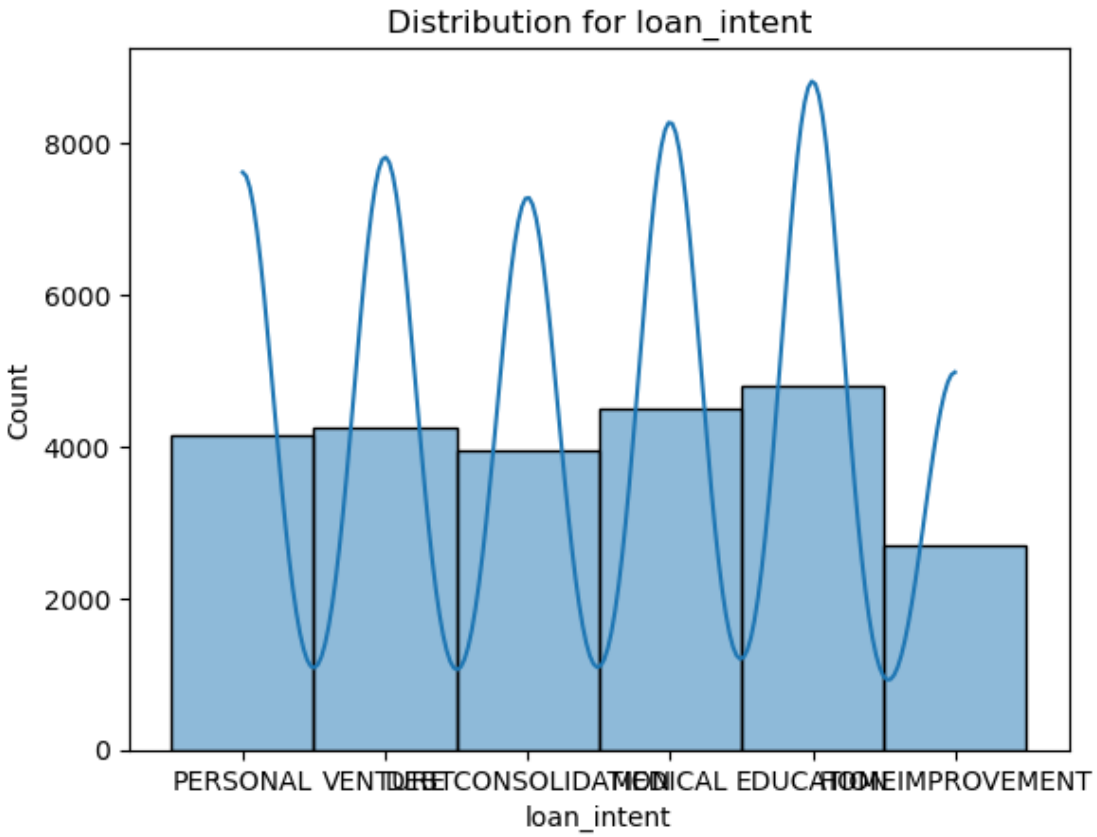
[829]:
```
for column in df.columns:
    sns.histplot(df[column].dropna(), kde=True)
    plt.title(f'Distribution for {column}')
    plt.show()
```

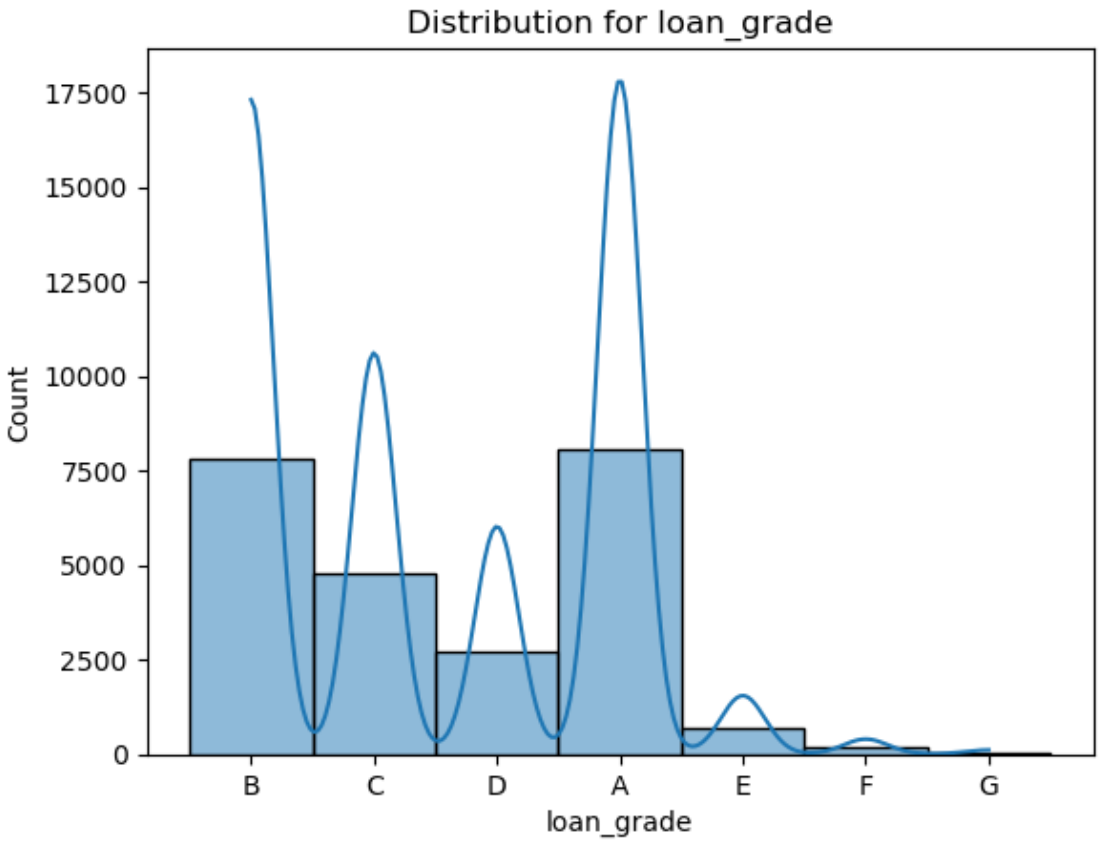

Distribution for person_age
Count
person_age
0
500
1000
1500
2000
2500
3000
20
40
60
80
100
120
140

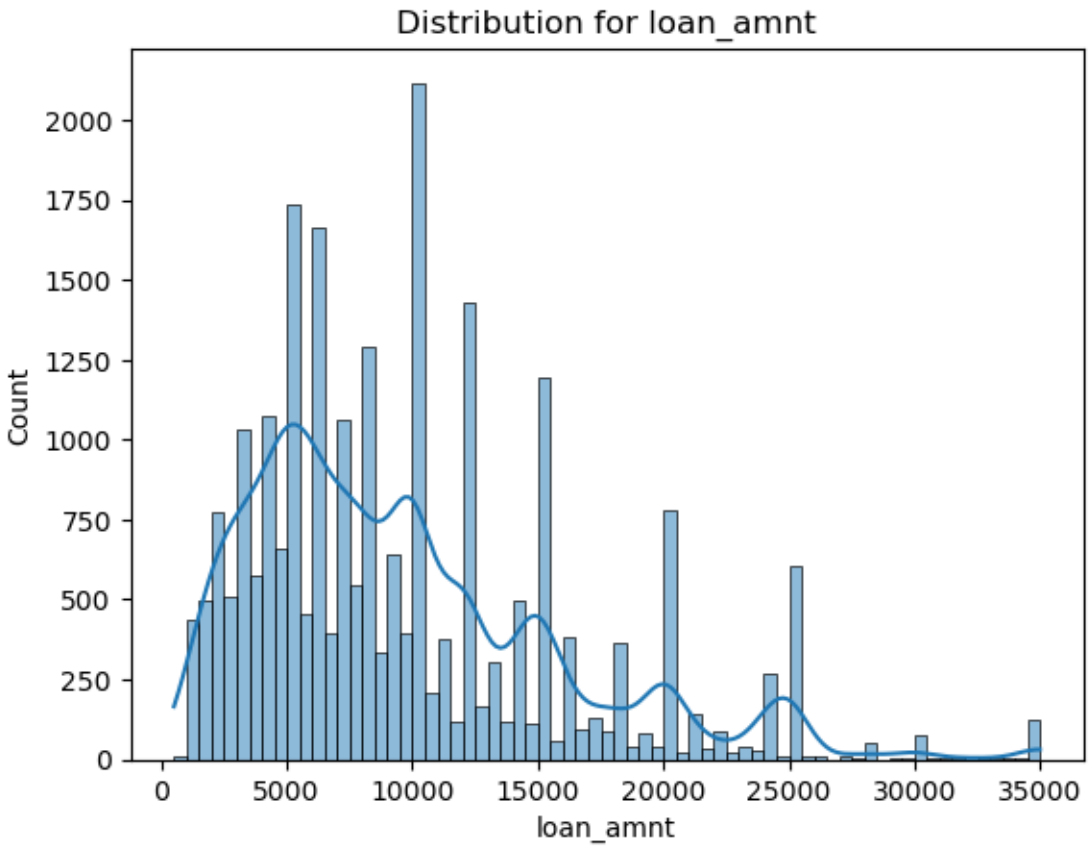

Distribution for person_income
Count
1200
1000
800
600
400
200
0
0
1
2
3
4
5
6
person_income
1e6

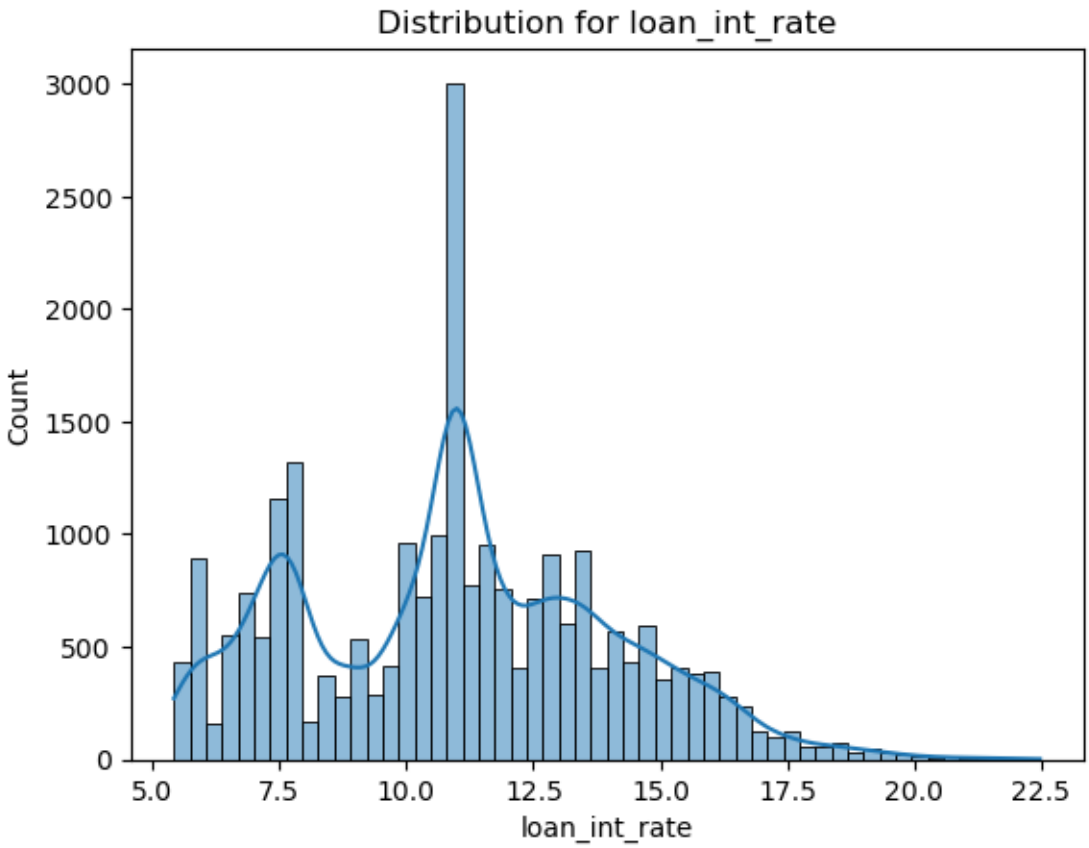

Distribution for person_home_ownership
60000
50000
40000
30000
20000
10000
0
Count
MORTGAGE
RENT
OWN
OTHER
person_home_ownership

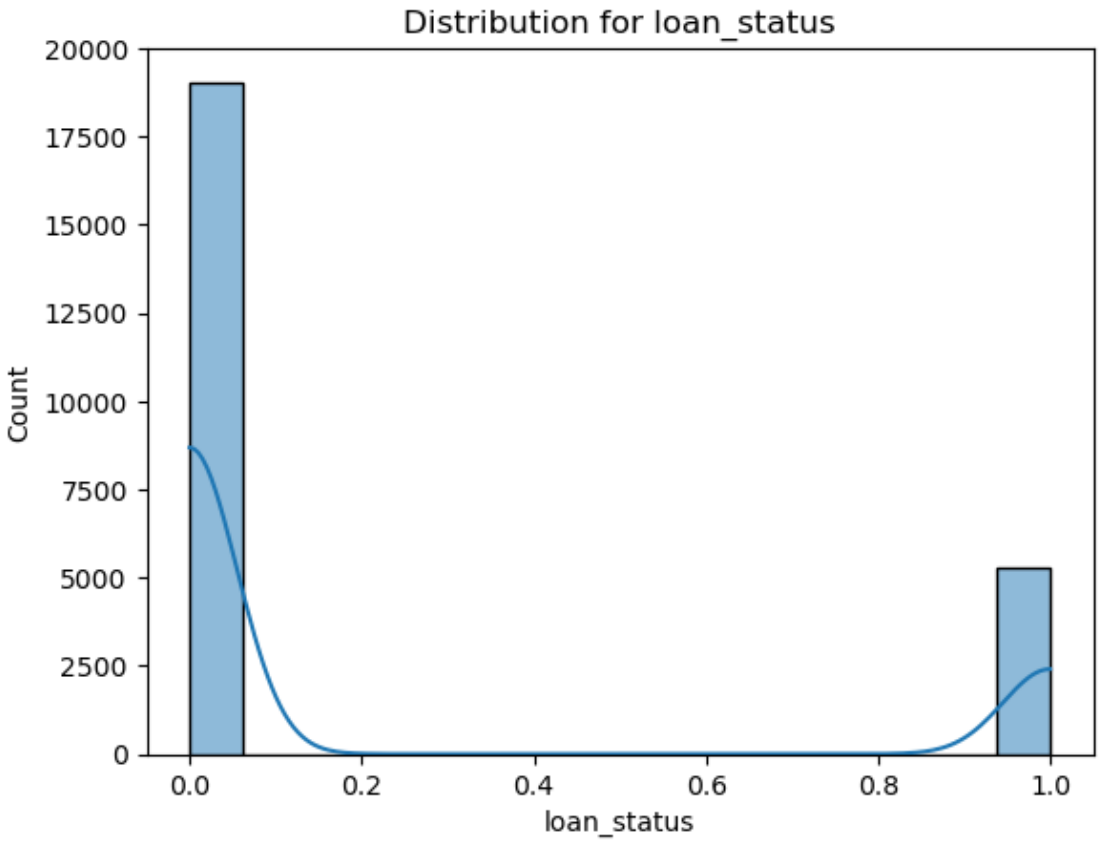

Distribution for person_emp_length
Count
3000
2500
2000
1500
1000
500
0
0
20
40
60
80
100
120
person_emp_length

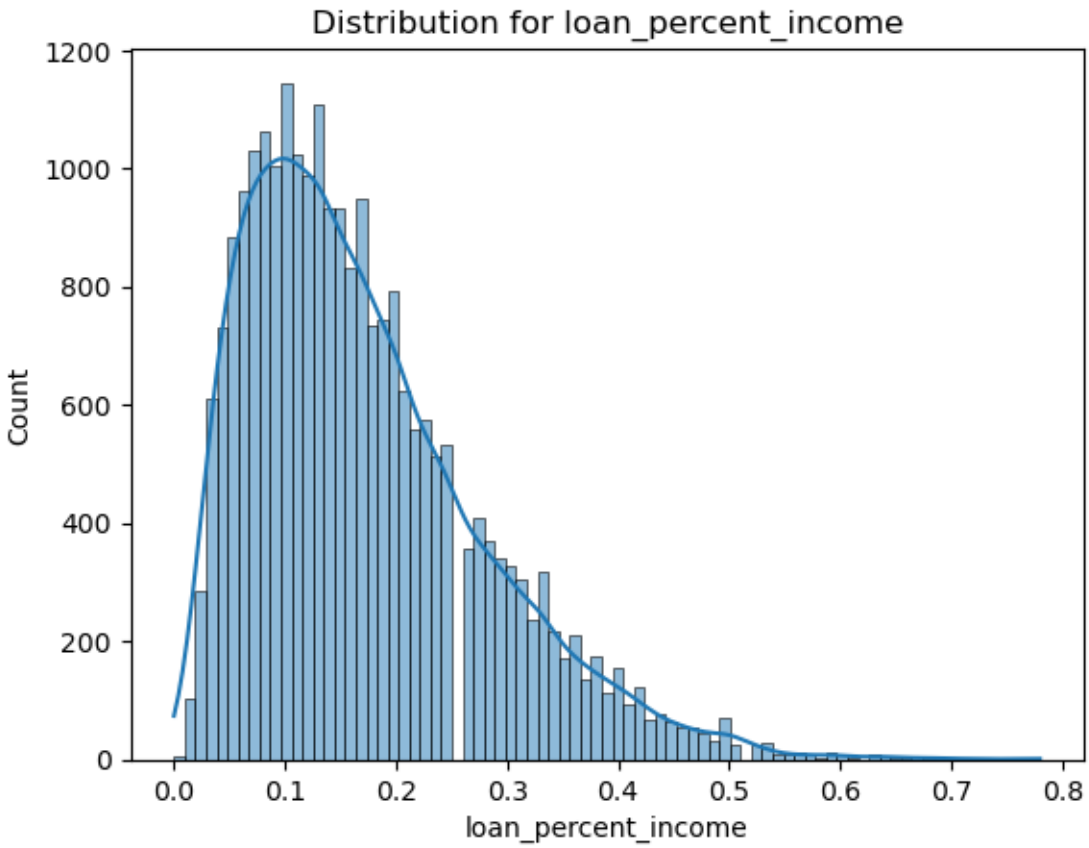

Distribution for loan_intent
Count
8000
6000
4000
2000
0
PERSONAL
VENTURE
DEBTCONSOLIDATION
MEDICAL
EDUCATION
HOMEIMPROVEMENT
loan_intent

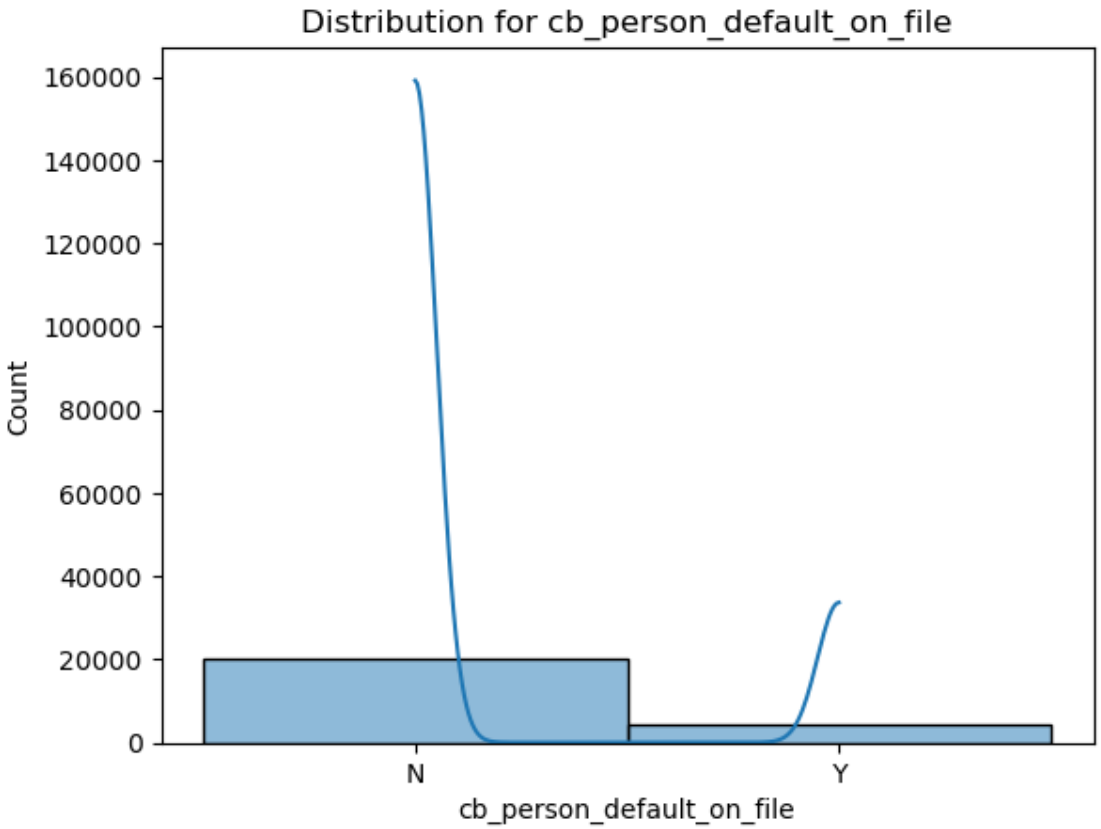

Distribution for loan_grade
Count
17500
15000
12500
10000
7500
5000
2500
0
B
C
D
A
E
F
G
loan_grade

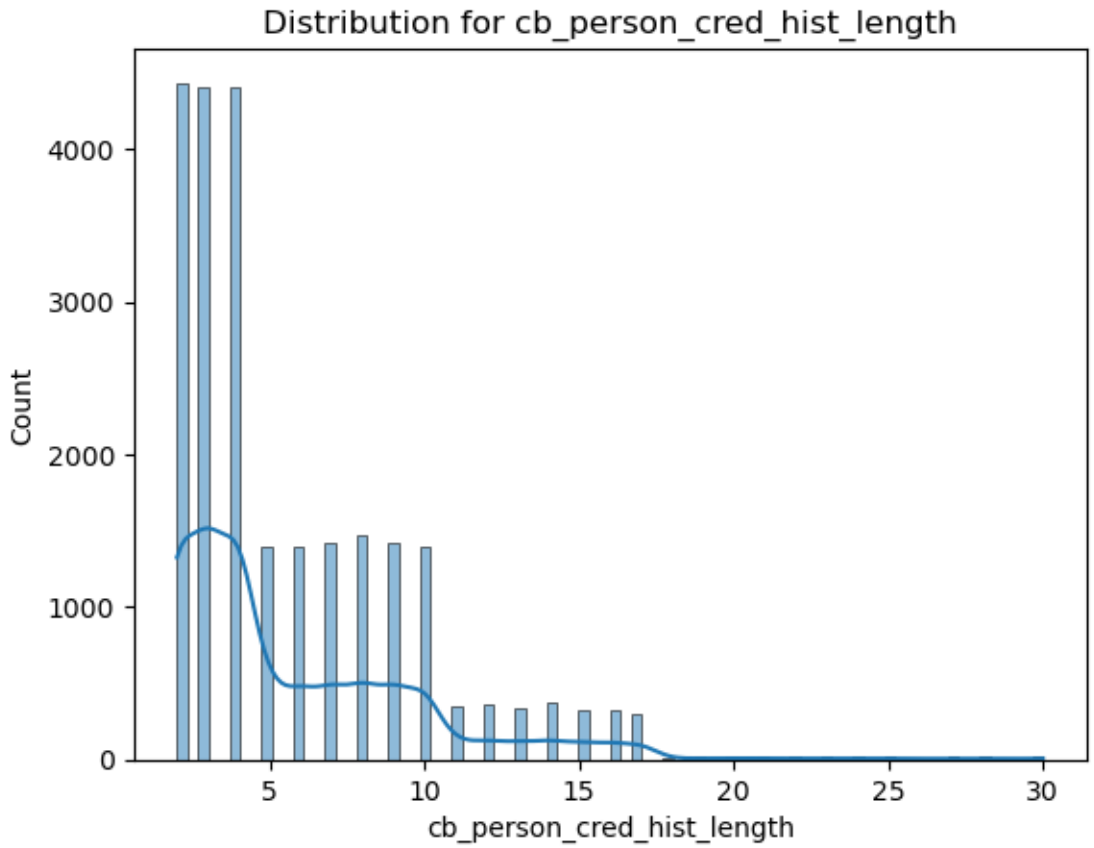

Distribution for loan_amnt
Count
loan_amnt
0
250
500
750
1000
1250
1500
1750
2000
0
5000
10000
15000
20000
25000
30000
35000


Distribution for loan_int_rate
Count
loan_int_rate
0
500
1000
1500
2000
2500
3000
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5

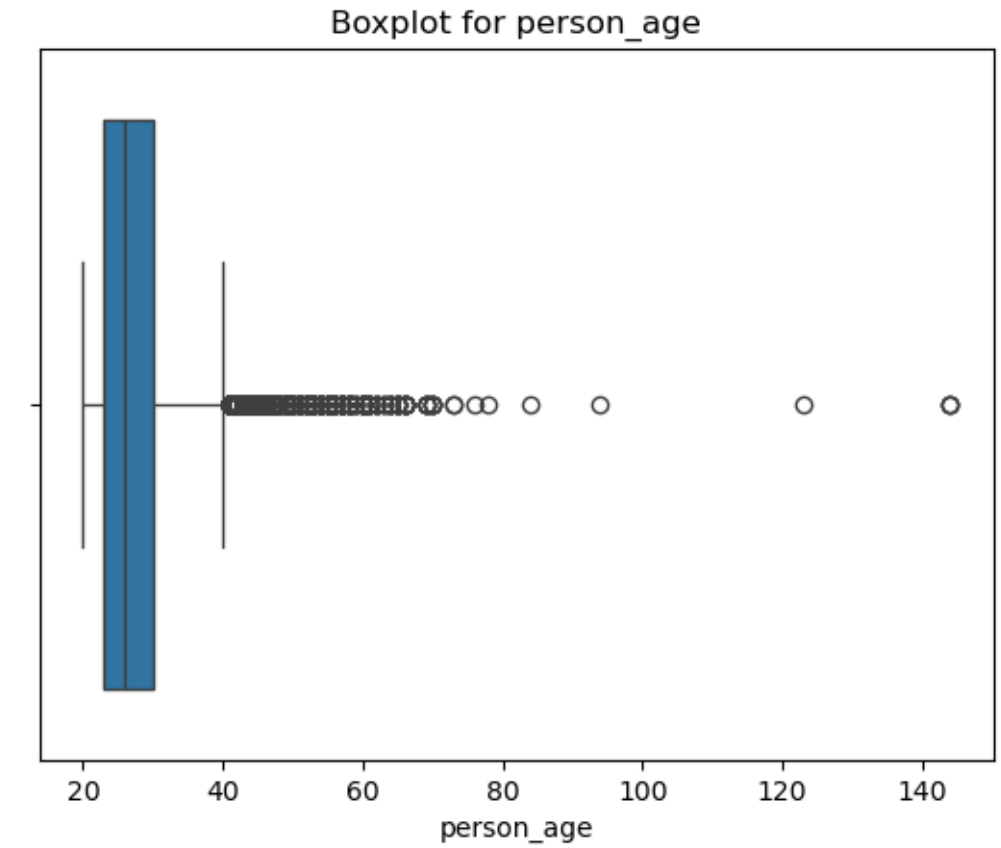

Distribution for loan_status
Count
loan_status
20000
17500
15000
12500
10000
7500
5000
2500
0
0.0
0.2
0.4
0.6
0.8
1.0

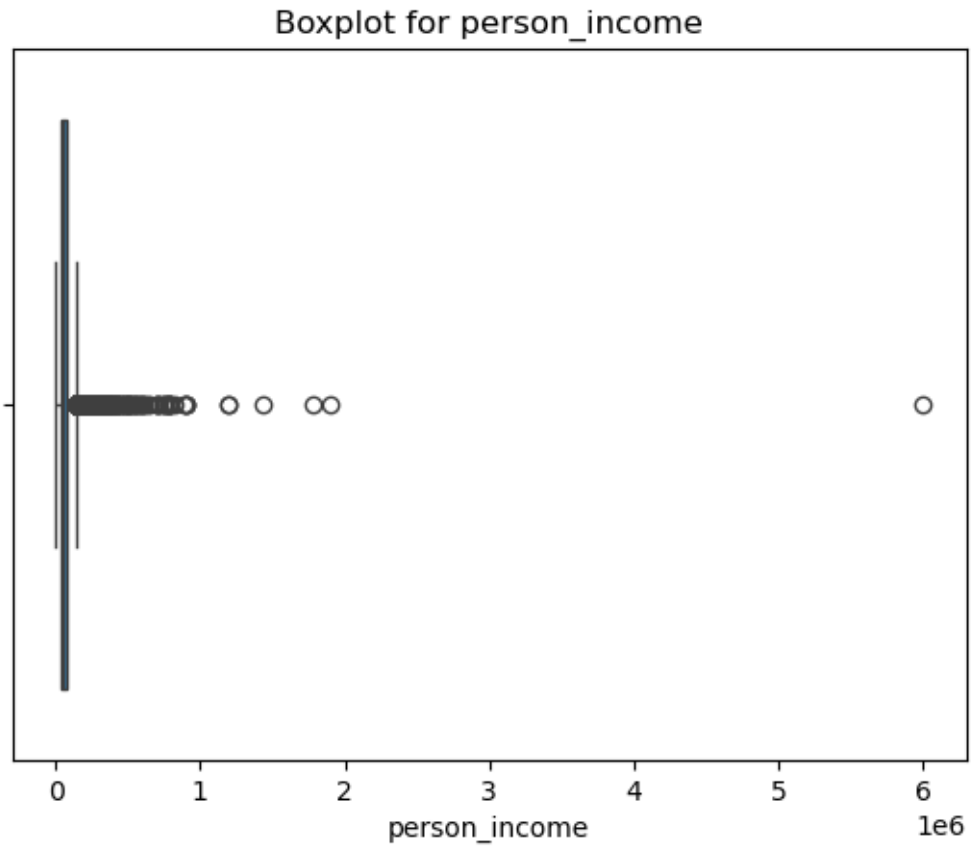

Distribution for loan_percent_income
Count
loan_percent_income
0
200
400
600
800
1000
1200
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

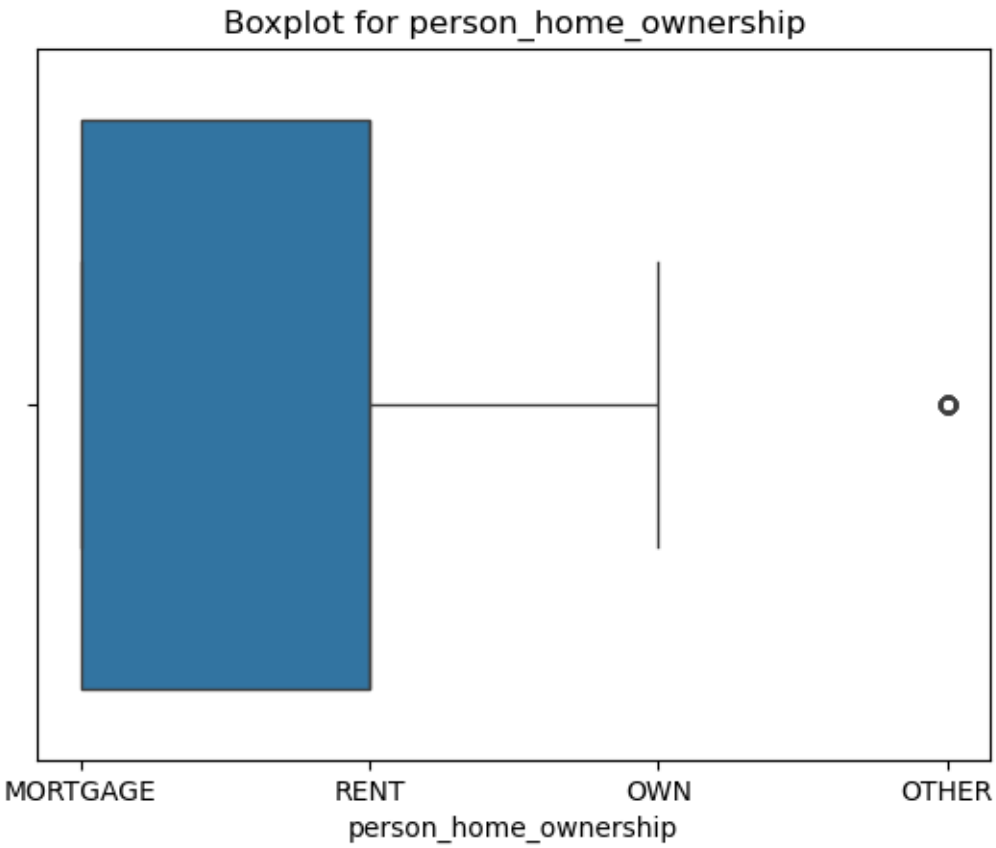

Distribution for cb_person_default_on_file
160000
140000
120000
100000
80000
60000
40000
20000
0
Count
N
Y
cb_person_default_on_file

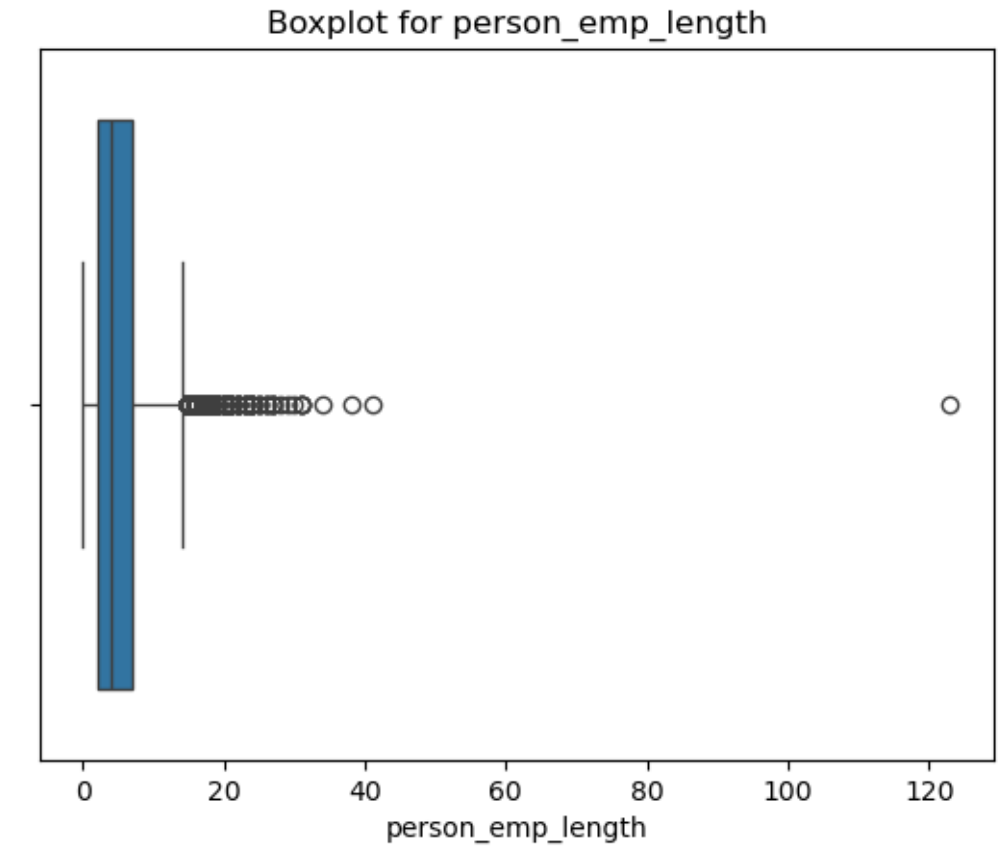


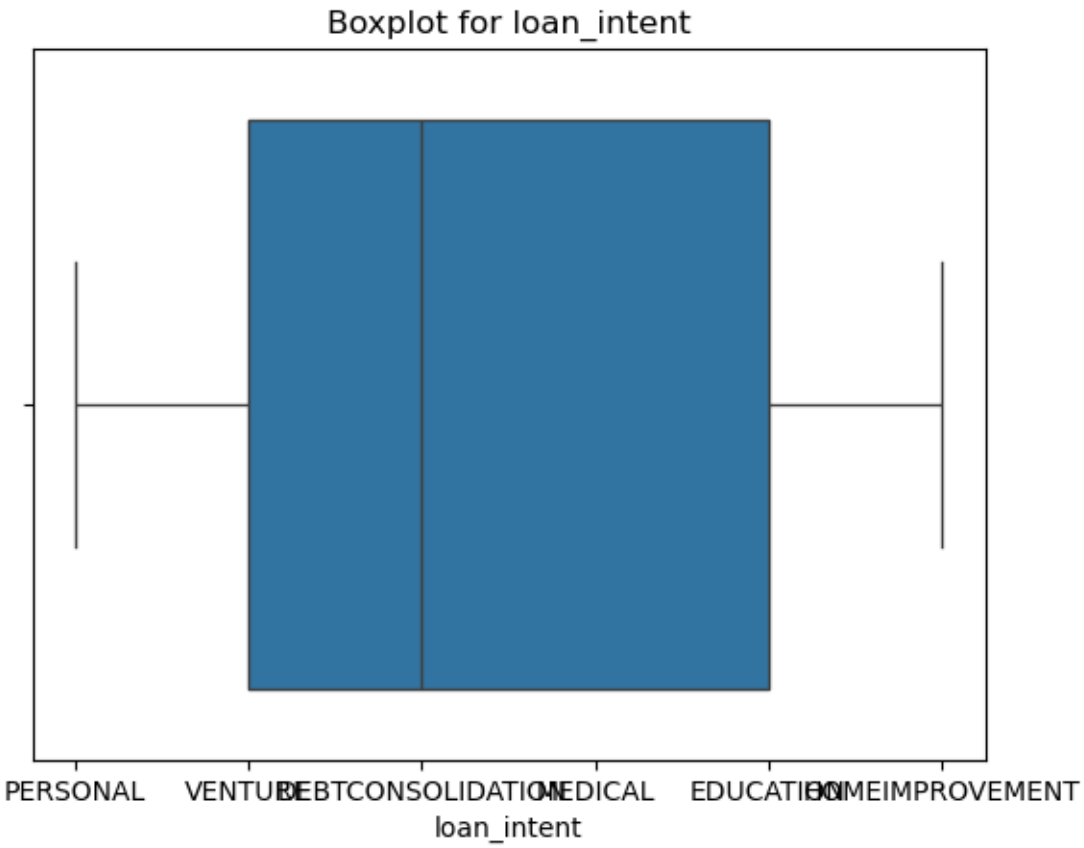
[830]:
```
for column in df.columns:
    sns.boxplot(data=df, x=column)
    plt.title(f'Boxplot for {column}')
    plt.show()
```

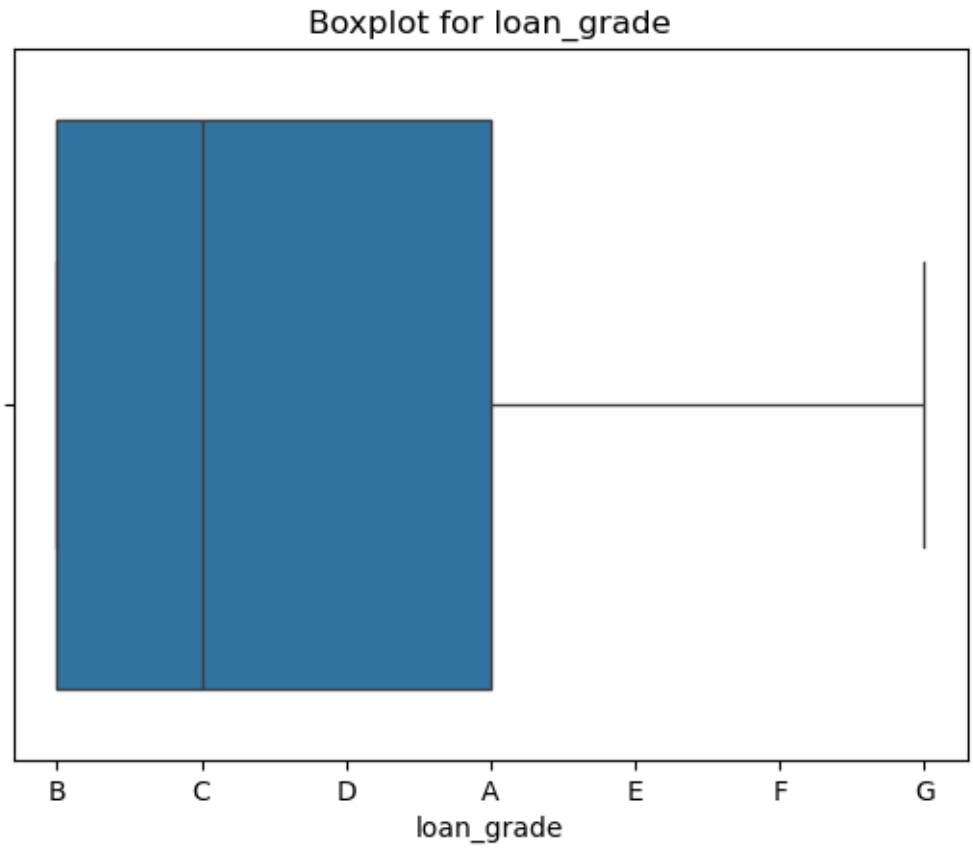

Boxplot for person_age
20
40
60
80
100
120
140
person_age

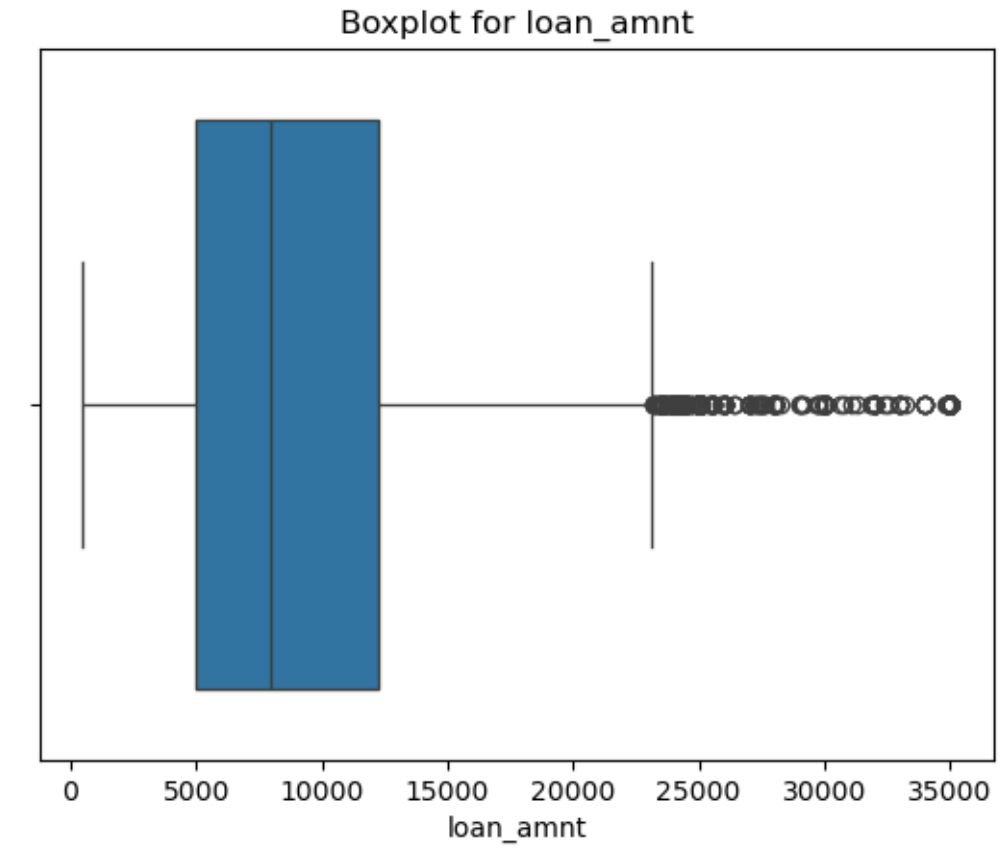

Boxplot for person_income
0
1
2
3
4
5
6
person_income
1e6

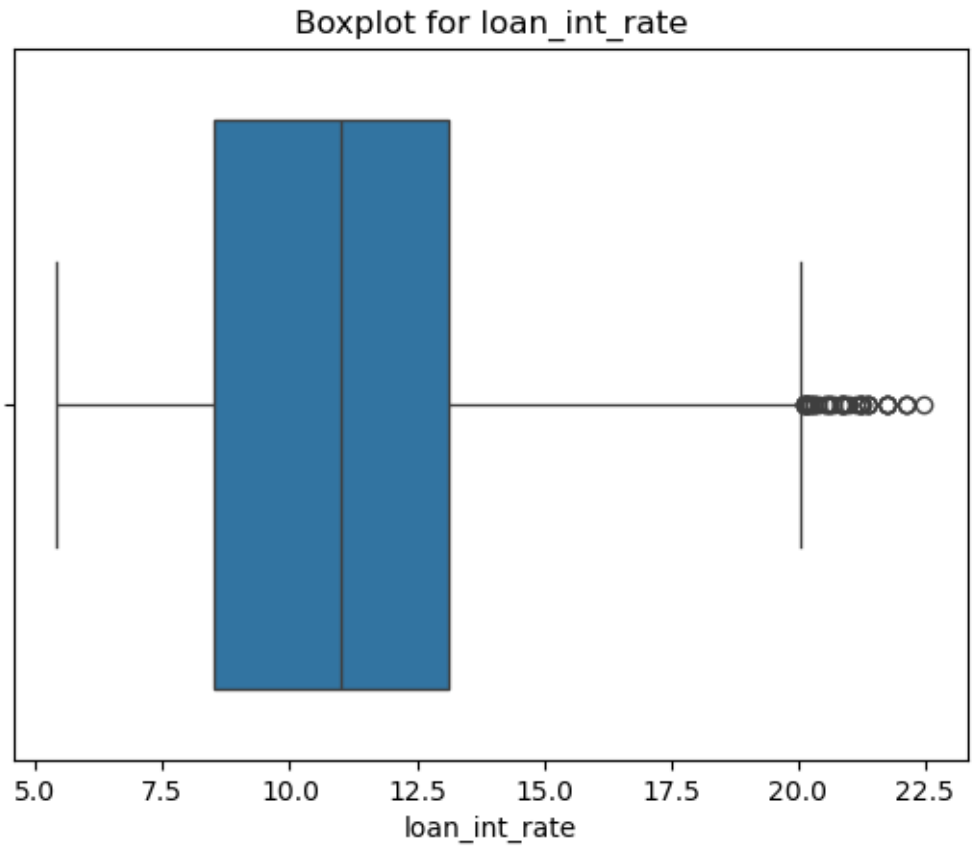

Boxplot for person_home_ownership
MORTGAGE
RENT
OWN
OTHER
person_home_ownership

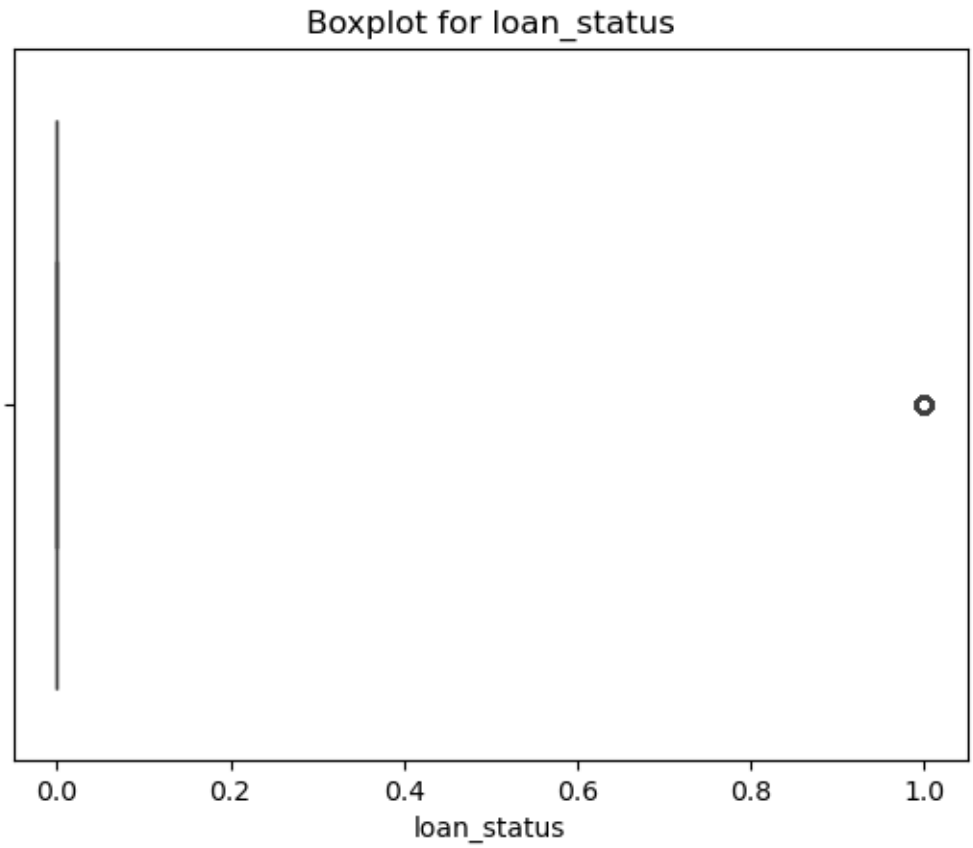

Boxplot for person_emp_length
0
20
40
60
80
100
120
person_emp_length

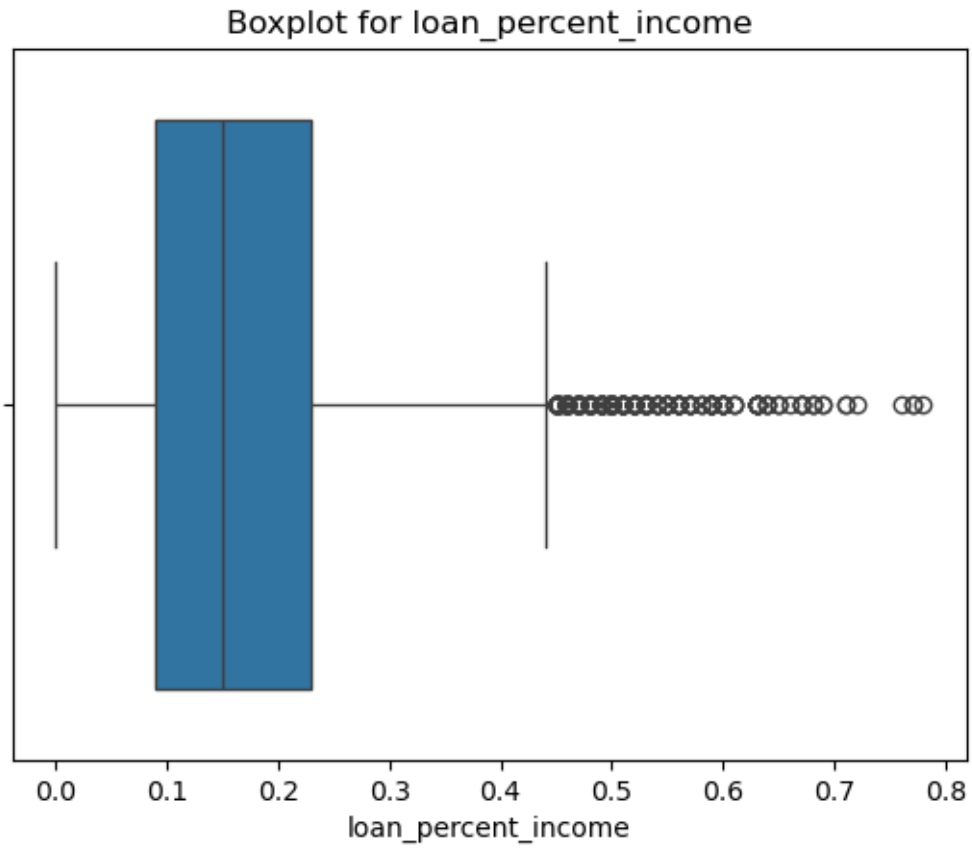

Boxplot for loan_intent
PERSONAL
VENTURE
DEBTCONSOLIDATION
MEDICAL
EDUCATION
HOMEIMPROVEMENT
loan_intent

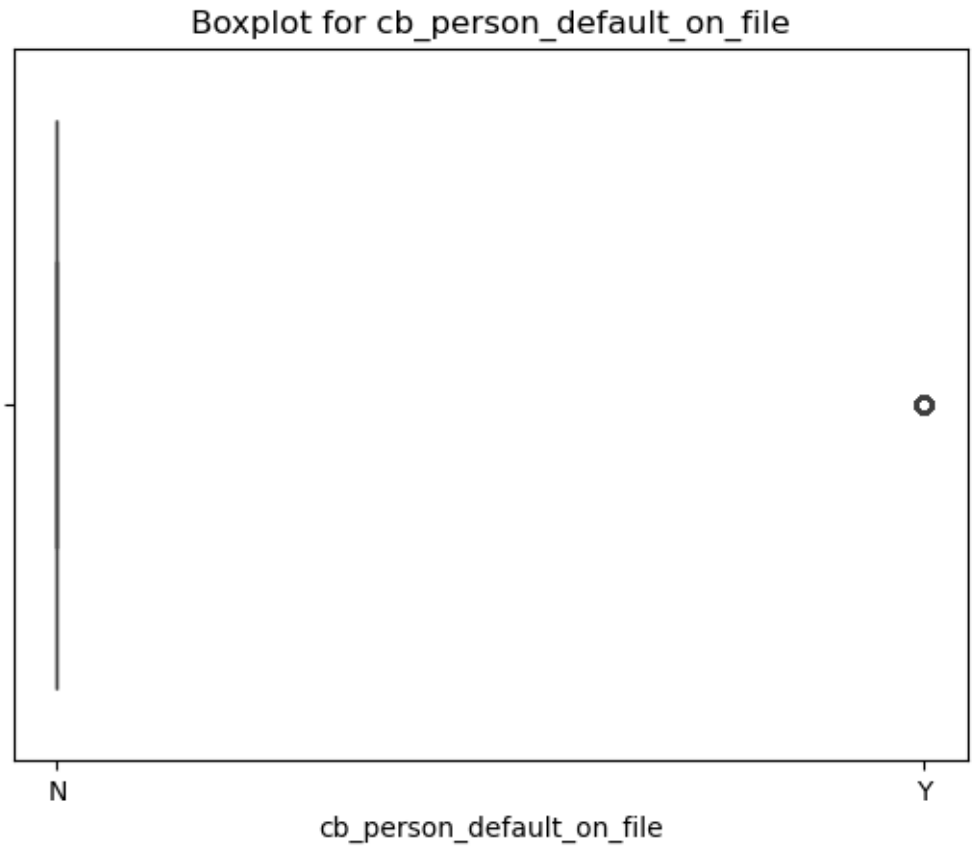

Boxplot for loan_grade
B
C
D
A
E
F
G
loan_grade

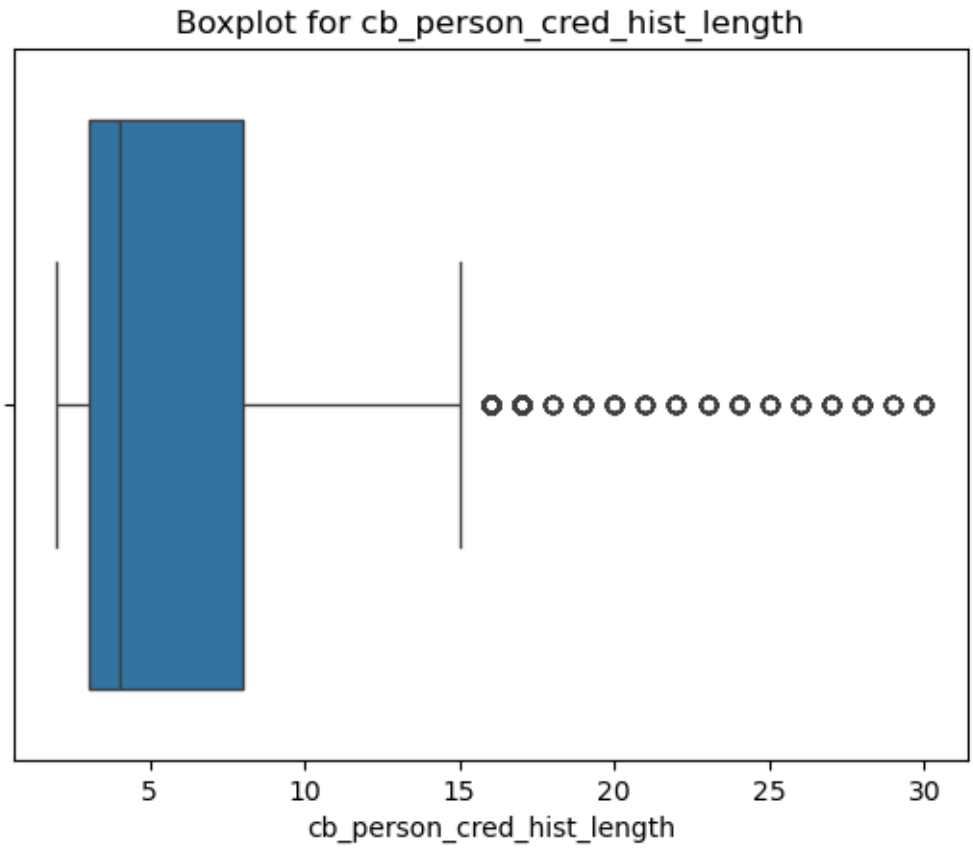

Boxplot for loan_amnt
0
5000
10000
15000
20000
25000
30000
35000
loan_amnt


Boxplot for loan_int_rate
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
loan_int_rate


Boxplot for loan_status
0.0
0.2
0.4
0.6
0.8
1.0
loan_status

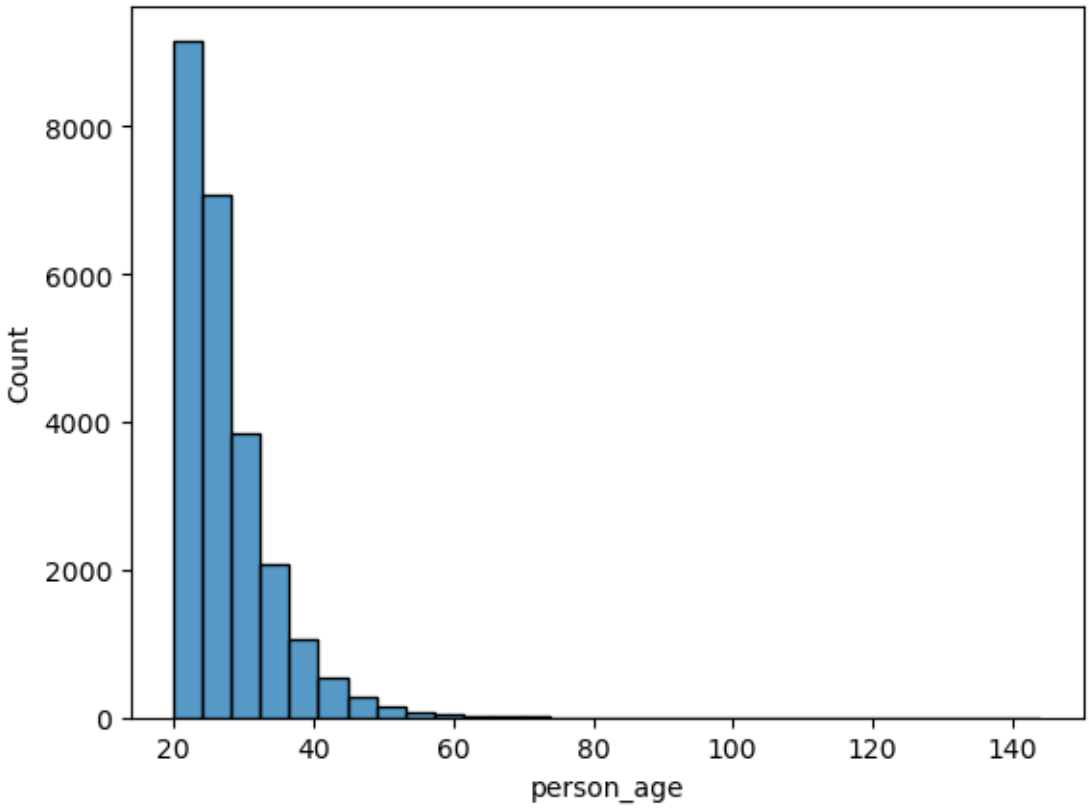

Boxplot for loan_percent_income
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
loan_percent_income


Boxplot for cb_person_default_on_file
N
Y
cb_person_default_on_file



[831]:
```
sns.histplot(df['person_age'], bins=30)
```

[831]:
```
<Axes: xlabel='person_age', ylabel='Count'>
```



We observe that the person_age variable includes clients aged 80+ and even over 100. These entries are clearly incorrect values, and since they are few in number, we have decided to remove them.

[833]:
```
df = df[df['person_age'] <= 80]
```

[834]:
```
df['person_income'].describe()
```

[834]:
```
count    2.432600e+04
mean     6.593198e+04
std      5.190047e+04
min      4.000000e+03
25%      3.900000e+04
50%      5.500000e+04
75%      7.943400e+04
max      1.900000e+06
Name: person_income, dtype: float64
```

[835]:
```
df[df['person_income'] > 1_000_000]
```

[835]:

	person_age	person_income	person_home_ownership	person_emp_length	\
4533	36	1200000	OWN	16.0	
11056	44	1440000	MORTGAGE	7.0	

```
19662          60        1900000              MORTGAGE               5.0
20730          32        1200000              MORTGAGE               1.0
21074          63        1782000                  RENT              13.0

            loan_intent loan_grade  loan_amnt  loan_int_rate  loan_status  \
4533            MEDICAL          A      10000           6.54            0
11056  DEBTCONSOLIDATION          A       6400           7.40            0
19662           PERSONAL          A       1500          10.99            0
20730            VENTURE          A      12000           7.51            0
21074          EDUCATION          C      12025          14.27            0

       loan_percent_income cb_person_default_on_file  \
4533                  0.01                        N
11056                 0.00                        N
19662                 0.00                        N
20730                 0.01                        N
21074                 0.01                        N

       cb_person_cred_hist_length
4533                           11
11056                          15
19662                          21
20730                           8
21074                          30
```

In the person_income column, we observe several clients with significantly higher income than the average. However, after analyzing other variables, we conclude that these values are valid, and we will create a "high income" flag to mark them.

[837]:
```
df['is_high_income'] = (df['person_income'] > 1_000_000).astype(int)
```

[838]:
```
df[df['person_emp_length'] > 30]
```

[838]:
```
       person_age  person_income person_home_ownership  person_emp_length  \
5584           53         106000              MORTGAGE               38.0
6185           22          59000                  RENT              123.0
8213           78          48000                  RENT               41.0
12158          48          70000              MORTGAGE               31.0
13094          47         178000                   OWN               31.0
14448          58          49000              MORTGAGE               34.0
20466          46         180000              MORTGAGE               31.0
22865          46         180000              MORTGAGE               31.0

           loan_intent loan_grade  loan_amnt  loan_int_rate  loan_status  \
5584          PERSONAL          B      20000           9.88            0
6185          PERSONAL          D      35000          16.02            1
8213           MEDICAL          A       3000           7.51            0
12158  HOMEIMPROVEMENT          D       9000          14.54            0
```

```
13094          VENTURE          B       9000          10.99            0
14448          MEDICAL          D       7500          13.55            1
20466          MEDICAL          B      18000           9.91            0
22865          VENTURE          B      18000           9.91            0

       loan_percent_income cb_person_default_on_file  \
5584                  0.19                         N
6185                  0.59                         Y
8213                  0.06                         N
12158                 0.13                         N
13094                 0.05                         N
14448                 0.15                         Y
20466                 0.10                         N
22865                 0.10                         N

       cb_person_cred_hist_length  is_high_income
5584                           23               0
6185                            3               0
8213                           25               0
12158                          17               0
13094                          17               0
14448                          24               0
20466                          12               0
22865                          11               0
```

We see that in the person_emp_length column, there is a client aged 22 with 123 years of work experience. This is clearly an anomaly, so we will replace this value with the median employment length for clients aged 22. The other high values appear to be valid and will be retained.

[840]:
```
median_emp_length_22 = df.loc[
    (df['person_age'] == 22) & (df['person_emp_length'] <= 100),
    'person_emp_length'
].median()

df.loc[
    (df['person_age'] == 22) & (df['person_emp_length'] > 100),
    'person_emp_length'
] = median_emp_length_22
```

[841]:
```
df.loc[6185]
```

[841]:
```
person_age                    22
person_income              59000
person_home_ownership       RENT
person_emp_length            4.0
loan_intent             PERSONAL
loan_grade                     D
loan_amnt                  35000
```

```
loan_int_rate                   16.02
loan_status                         1
loan_percent_income              0.59
cb_person_default_on_file           Y
cb_person_cred_hist_length          3
is_high_income                      0
Name: 6185, dtype: object
```

[842]:
```
df['loan_amnt'].describe()
```

[842]:
```
count    24326.000000
mean      9589.160774
std       6303.199289
min        500.000000
25%       5000.000000
50%       8000.000000
75%      12250.000000
max      35000.000000
Name: loan_amnt, dtype: float64
```

[843]:
```
df['loan_int_rate'].describe()
```

[843]:
```
count    24326.000000
mean        11.000879
std          3.072789
min          5.420000
25%          8.490000
50%         10.990000
75%         13.110000
max         22.480000
Name: loan_int_rate, dtype: float64
```

[844]:
```
df[df['loan_percent_income'] > 0.5]
```

[844]:

	person_age	person_income	person_home_ownership	person_emp_length	\
41	22	36000	RENT	6.0	
231	29	8500	RENT	7.0	
305	27	13500	RENT	2.0	
347	23	65500	RENT	4.0	
356	24	43896	RENT	8.0	
…	…	…	…	…	
23228	23	23000	RENT	3.0	
23261	32	12000	RENT	2.0	
23348	31	39000	MORTGAGE	5.0	
23628	29	40000	MORTGAGE	13.0	
23834	23	40800	OWN	4.0	

```
             loan_intent loan_grade  loan_amnt  loan_int_rate  loan_status  \
41     DEBTCONSOLIDATION          B      18200          11.71            1
231              VENTURE          A       4500           8.63            1
305              MEDICAL          B       8000          12.18            1
347              MEDICAL          C      35000          15.23            1
356    HOMEIMPROVEMENT          B      23975          10.99            1
…                   …        …         …             …          …
23228         EDUCATION          C      12700          14.27            1
23261         EDUCATION          A       6200           6.17            1
23348         EDUCATION          B      22000          11.36            1
23628    HOMEIMPROVEMENT          B      25000          11.86            0
23834            MEDICAL          A      21000           6.17            0

       loan_percent_income cb_person_default_on_file  \
41                    0.51                         N
231                   0.53                         N
305                   0.59                         N
347                   0.53                         N
356                   0.55                         N
…                     …                        …
23228                 0.55                         Y
23261                 0.52                         N
23348                 0.56                         N
23628                 0.63                         N
23834                 0.51                         N

       cb_person_cred_hist_length  is_high_income
41                              3               0
231                            10               0
305                             5               0
347                             2               0
356                             2               0
…                             …               …
23228                           4               0
23261                           9               0
23348                           5               0
23628                           6               0
23834                           3               0

[171 rows x 13 columns]
```

[845]:
```
df[
    (df['cb_person_cred_hist_length'] > 15) &
    (df['person_age'] < 30)
]
```

[845]:
```
Empty DataFrame
Columns: [person_age, person_income, person_home_ownership, person_emp_length,
loan_intent, loan_grade, loan_amnt, loan_int_rate, loan_status,
loan_percent_income, cb_person_default_on_file, cb_person_cred_hist_length,
is_high_income]
Index: []
```

[846]:
```
df.describe()
```

[846]:

	person_age	person_income	person_emp_length	loan_amnt
count	24326.000000	2.432600e+04	24326.000000	24326.000000
mean	27.743525	6.593198e+04	4.755036	9589.160774
std	6.217010	5.190047e+04	3.986881	6303.199289
min	20.000000	4.000000e+03	0.000000	500.000000
25%	23.000000	3.900000e+04	2.000000	5000.000000
50%	26.000000	5.500000e+04	4.000000	8000.000000
75%	30.000000	7.943400e+04	7.000000	12250.000000
max	78.000000	1.900000e+06	41.000000	35000.000000

	loan_int_rate	loan_status	loan_percent_income
count	24326.000000	24326.000000	24326.000000
mean	11.000879	0.217052	0.170180
std	3.072789	0.412247	0.106772
min	5.420000	0.000000	0.000000
25%	8.490000	0.000000	0.090000
50%	10.990000	0.000000	0.150000
75%	13.110000	0.000000	0.230000
max	22.480000	1.000000	0.780000

	cb_person_cred_hist_length	is_high_income
count	24326.000000	24326.000000
mean	5.816698	0.000206
std	4.045444	0.014336
min	2.000000	0.000000
25%	3.000000	0.000000
50%	4.000000	0.000000
75%	8.000000	0.000000
max	30.000000	1.000000

[847]:
```
df['cb_person_default_on_file'].describe()
```

[847]:
```
count     24326
unique        2
top           N
freq      20079
Name: cb_person_default_on_file, dtype: object
```

[848]:
```
df
```

[848]:

	person_age	person_income	person_home_ownership	person_emp_length	\
0	26	200100	MORTGAGE	10.0	
1	31	64450	RENT	3.0	
2	22	22292	RENT	5.0	
3	33	109000	MORTGAGE	4.0	
4	24	100000	MORTGAGE	8.0	
…	…	…	…	…	
24430	39	38500	MORTGAGE	7.0	
24431	25	69000	RENT	5.0	
24432	26	148000	RENT	1.0	
24433	26	175000	MORTGAGE	0.0	
24434	27	65000	MORTGAGE	11.0	

	loan_intent	loan_grade	loan_amnt	loan_int_rate	loan_status	\
0	PERSONAL	B	30000	11.71	0	
1	VENTURE	C	12000	12.73	0	
2	DEBTCONSOLIDATION	D	3500	16.49	1	
3	MEDICAL	B	15000	11.49	0	
4	EDUCATION	A	16800	7.88	0	
…	…	…	…	…	…	
24430	MEDICAL	C	3500	13.98	0	
24431	HOMEIMPROVEMENT	A	8500	6.92	1	
24432	DEBTCONSOLIDATION	E	20000	17.99	1	
24433	PERSONAL	C	15000	10.99	0	
24434	VENTURE	A	8000	7.49	0	

	loan_percent_income	cb_person_default_on_file	\
0	0.15	N	
1	0.19	N	
2	0.16	N	
3	0.14	N	
4	0.17	N	
…	…	…	
24430	0.09	Y	
24431	0.12	N	
24432	0.14	N	
24433	0.09	N	
24434	0.12	N	

	cb_person_cred_hist_length	is_high_income
0	4	0
1	8	0
2	3	0
3	5	0
4	4	0

```
…                              …              …
24430                             17              0
24431                              4              0
24432                              3              0
24433                              3              0
24434                              8              0

[24326 rows x 13 columns]
```

Thus, incorrect outlier values have been handled, while valid outliers have been retained for further analysis. During the modeling stage, we will apply techniques to mitigate their influence.

Duplicates, missing, and incorrect values have been handled — the data is now ready for further processing.

[852]:
```
df_cleared = df
df_cleared.to_csv('credit_risk_cleared_data.csv', index=False)
```

Researching analysis

[854]:
```
df.describe()
```

[854]:
```
         person_age  person_income  person_emp_length     loan_amnt  \
count  24326.000000   2.432600e+04       24326.000000  24326.000000
mean      27.743525   6.593198e+04           4.755036   9589.160774
std        6.217010   5.190047e+04           3.986881   6303.199289
min       20.000000   4.000000e+03           0.000000    500.000000
25%       23.000000   3.900000e+04           2.000000   5000.000000
50%       26.000000   5.500000e+04           4.000000   8000.000000
75%       30.000000   7.943400e+04           7.000000  12250.000000
max       78.000000   1.900000e+06          41.000000  35000.000000

       loan_int_rate   loan_status  loan_percent_income  \
count   24326.000000  24326.000000         24326.000000
mean       11.000879      0.217052             0.170180
std         3.072789      0.412247             0.106772
min         5.420000      0.000000             0.000000
25%         8.490000      0.000000             0.090000
50%        10.990000      0.000000             0.150000
75%        13.110000      0.000000             0.230000
max        22.480000      1.000000             0.780000

       cb_person_cred_hist_length  is_high_income
count                24326.000000    24326.000000
mean                     5.816698        0.000206
std                      4.045444        0.014336
min                      2.000000        0.000000
25%                      3.000000        0.000000
50%                      4.000000        0.000000
```

```
75%                    8.000000        0.000000
max                   30.000000        1.000000
```

Financial Profile:

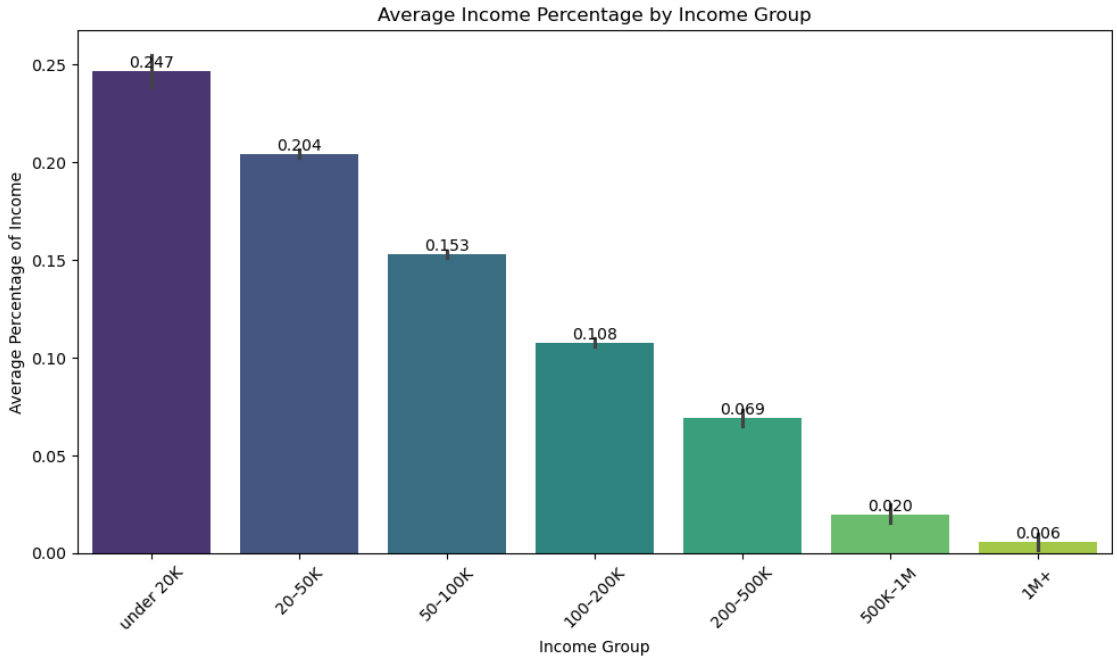
What is the average loan_percent_income across different income levels?

[860]:
```
bins = [0, 20000, 50000, 100000, 200000, 500000, 1000000, float('inf')]
labels = ['under 20K', '20-50K', '50-100K', '100-200K', '200-500K', '500K-1M',␣
 ↪'1M+']

df['income_group'] = pd.cut(df['person_income'], bins=bins, labels=labels)

grouped = df.groupby('income_group', observed=True)['loan_percent_income'].
 ↪agg(['mean', 'count']).reset_index()
grouped.rename(columns={'mean': 'avg_loan_percent_income', 'count': 'clients'},␣
 ↪inplace=True)

grouped
```

[860]:

	income_group	avg_loan_percent_income	clients
0	under 20K	0.246507	876
1	20-50K	0.204398	9613
2	50-100K	0.152907	10684
3	100-200K	0.107863	2827
4	200-500K	0.068982	285
5	500K-1M	0.019722	36
6	1M+	0.006000	5

[862]:
```
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=df, x='income_group', y='loan_percent_income',␣
 ↪hue='income_group', palette='viridis')

for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.3f}',
                (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom', fontsize=10)

plt.title('Average Income Percentage by Income Group')
plt.xlabel('Income Group')
plt.ylabel('Average Percentage of Income')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

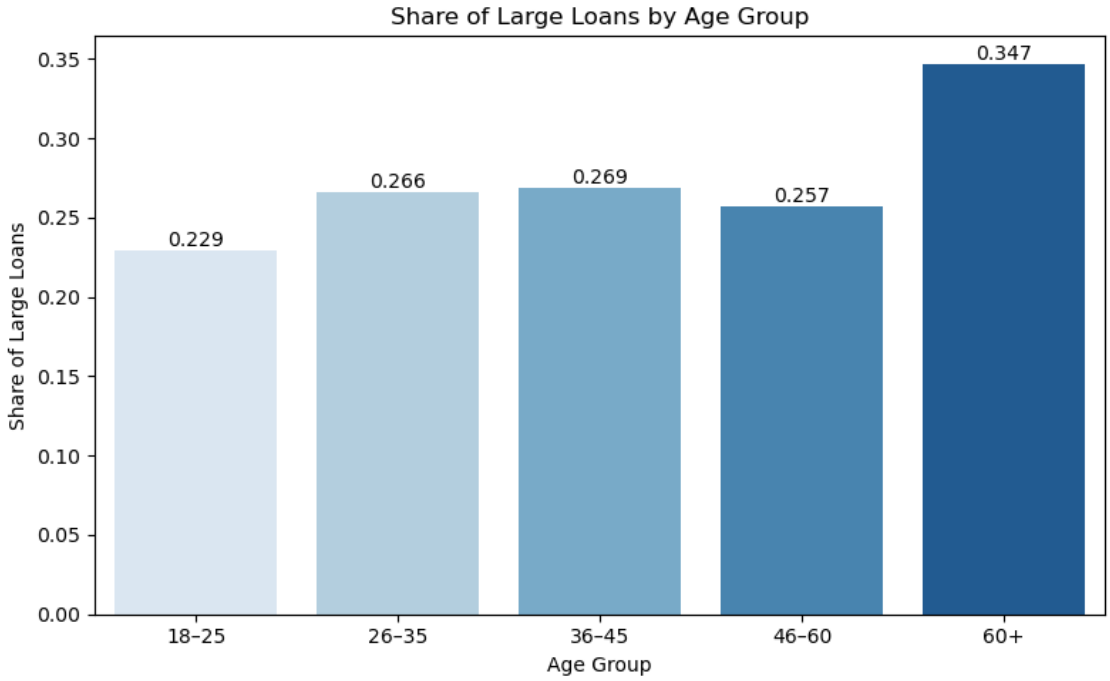


Who most often takes out large loans — by age, experience, property type?

[872]:
```
high_loan_threshold = df['loan_amnt'].quantile(0.75) # large loan
df['is_large_loan'] = (df['loan_amnt'] > high_loan_threshold).astype(int)

df['age_group'] = pd.cut(df['person_age'], bins=[17, 25, 35, 45, 60, 80],
                         labels=['18-25', '26-35', '36-45', '46-60', '60+'])
age_analysis = df.groupby('age_group', observed=True)['is_large_loan'].mean().
 ↪reset_index()

df['work_group'] = pd.cut(df['person_emp_length'], bins=[-1, 2, 5, 10, 20, 100],
                          labels=['0-2   ', '3-5   ', '6-10   ', '11-20   ',␣
 ↪'20+   '])
emp_analysis = df.groupby('work_group', observed=True)['is_large_loan'].mean().
 ↪reset_index()

ownership_analysis = df.groupby('person_home_ownership')['is_large_loan'].
 ↪mean().reset_index()
```

[877]:
```
plt.figure(figsize=(8, 5))
ax = sns.barplot(data=age_analysis, x='age_group', y='is_large_loan',␣
 ↪hue='age_group', palette='Blues')

for p in ax.patches:
    height = p.get_height()
```

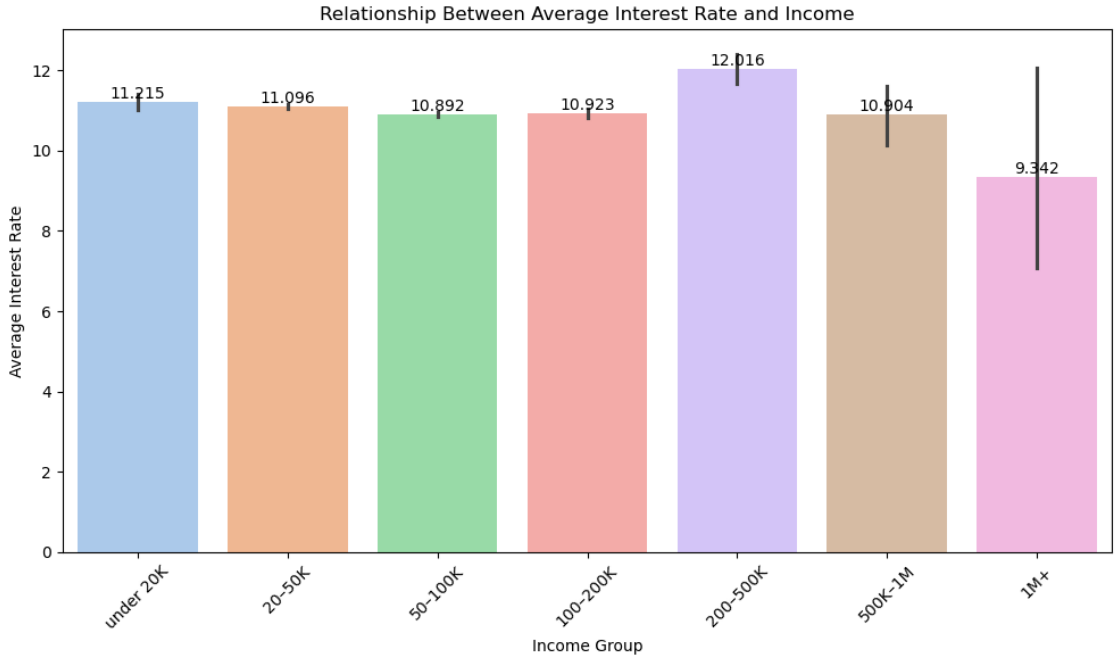
```
    ax.annotate(f'{height:.3f}',
                (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom', fontsize=10, color='black')

plt.title('Share of Large Loans by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Share of Large Loans')
plt.tight_layout()
plt.show()
```



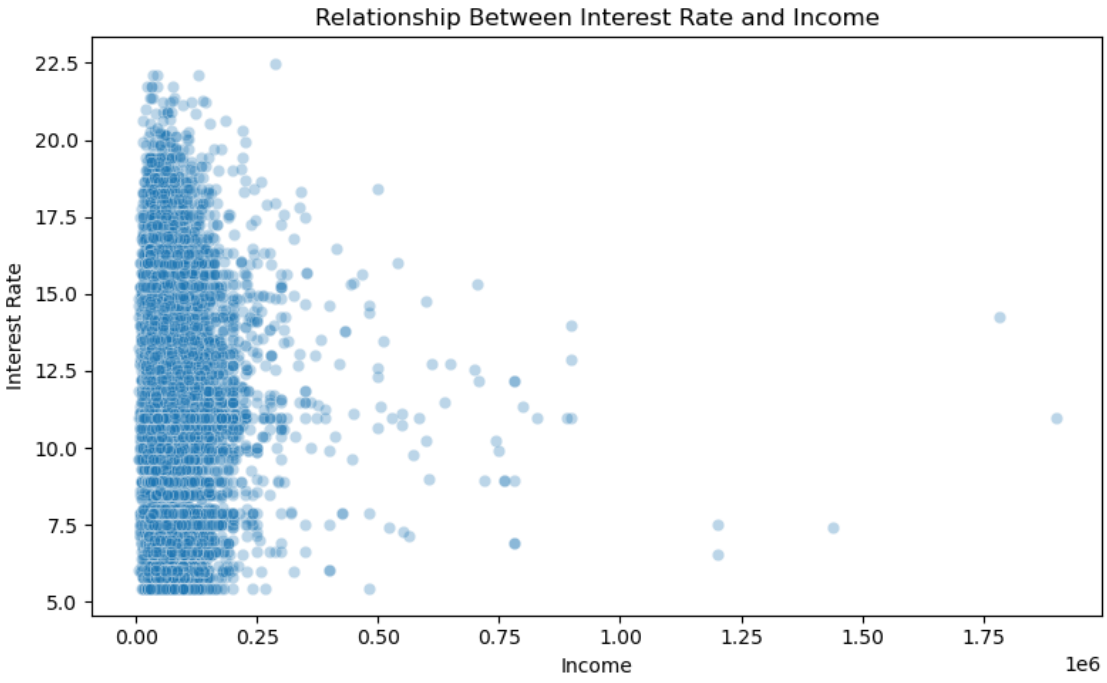
How are income and interest rate (loan_int_rate) related?

[884]:
```
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=df, x='income_group', y='loan_int_rate',␣
 ↪hue='income_group', palette='pastel')

for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.3f}',
                (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom', fontsize=10)

plt.title('Relationship Between Average Interest Rate and Income')
plt.xlabel('Income Group')
plt.ylabel('Average Interest Rate')
```

```
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

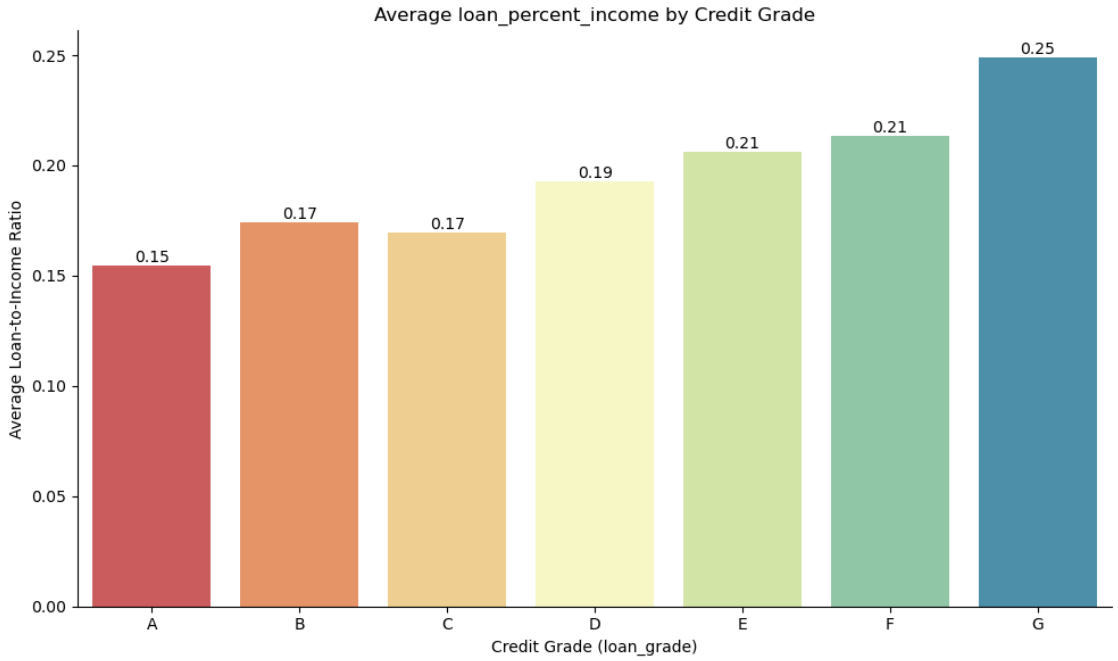


[885]:
```
plt.figure(figsize=(8, 5))
sns.scatterplot(data=df, x='person_income', y='loan_int_rate', alpha=0.3)
plt.title('Relationship Between Interest Rate and Income')
plt.xlabel('Income')
plt.ylabel('Interest Rate')
plt.tight_layout()
plt.show()
```



How does loan_percent_income vary depending on credit grade (loan_grade)?

[889]:
```
grade_lpi = df.groupby('loan_grade', observed=True)['loan_percent_income'].
 ↪agg(['mean', 'count']).reset_index()
grade_lpi.rename(columns={'mean': 'avg_loan_percent_income', 'count':␣
 ↪'clients'}, inplace=True)

plt.figure(figsize=(10, 6))
ax = sns.barplot(data=grade_lpi, x='loan_grade', y='avg_loan_percent_income',␣
 ↪hue='loan_grade', palette='Spectral')

for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.2f}',
                (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom', fontsize=10)

plt.title('Average loan_percent_income by Credit Grade')
plt.xlabel('Credit Grade (loan_grade)')
plt.ylabel('Average Loan-to-Income Ratio')
sns.despine()
plt.tight_layout()
plt.show()
```



Credit History and Risk:

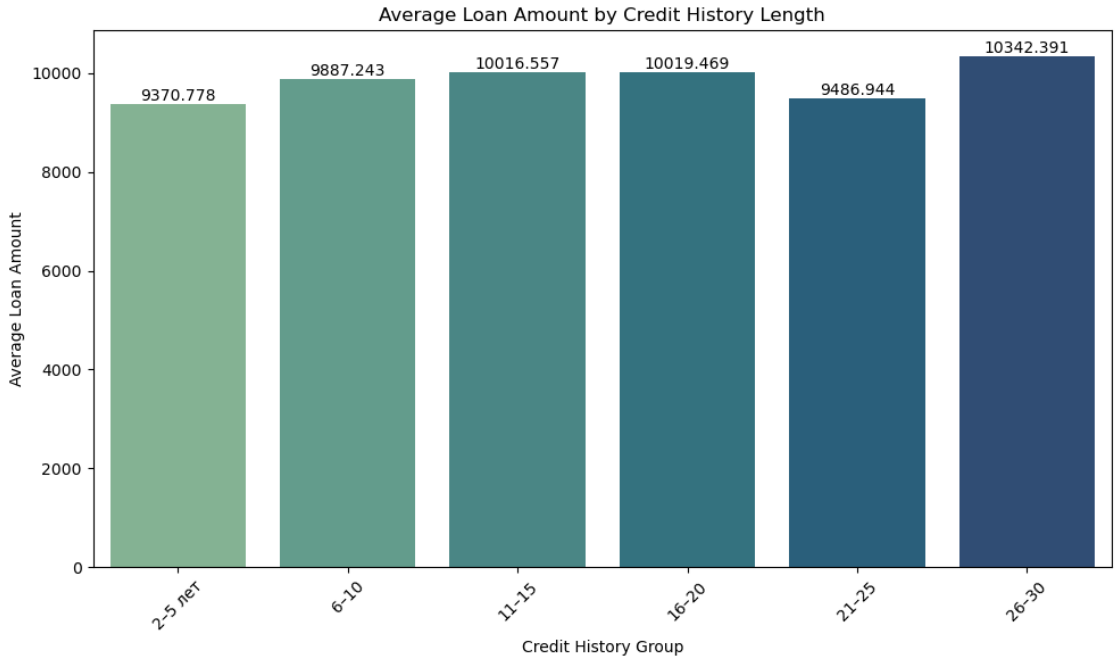
How does credit history length (cred_hist_length) affect loan amount?

[892]:
```
df['cred_hist_group'] = pd.cut(
    df['cb_person_cred_hist_length'],
    bins=[1, 5, 10, 15, 20, 25, 30],
    labels=['2-5   ', '6-10', '11-15', '16-20', '21-25', '26-30']
)

grouped = df.groupby('cred_hist_group', observed=True)['loan_amnt'].
 ↪agg(['mean', 'count']).reset_index()
grouped.rename(columns={'mean': 'avg_loan_amnt', 'count': 'clients'},␣
 ↪inplace=True)

grouped
```

[892]:

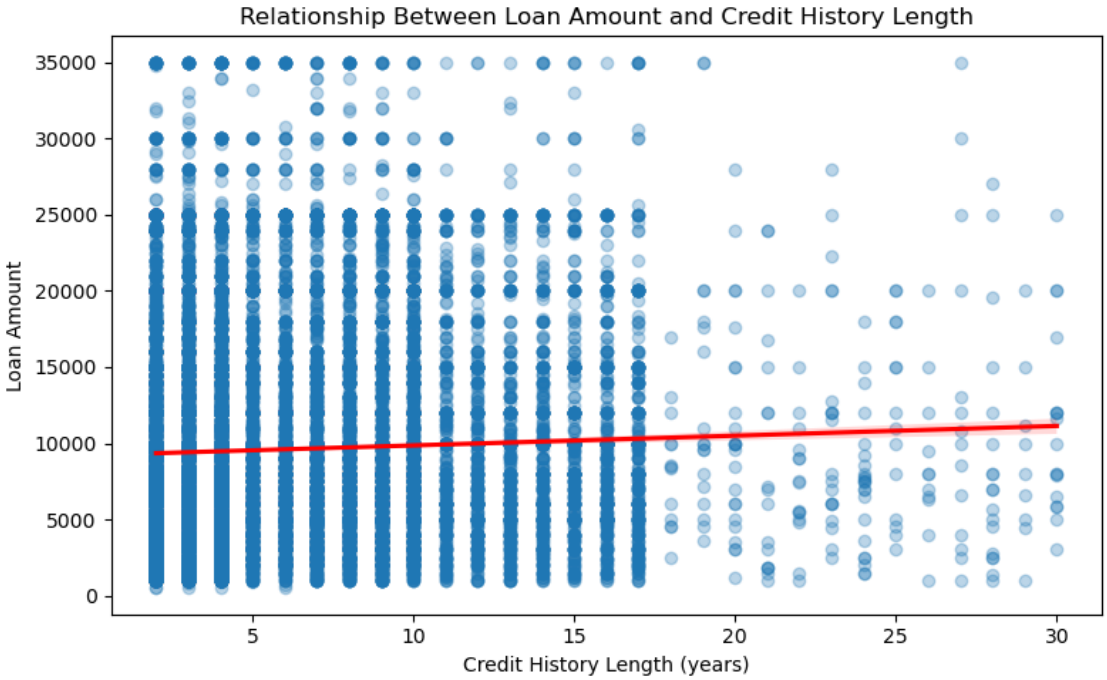
	cred_hist_group	avg_loan_amnt	clients
0	2-5	9370.777558	14642
1	6-10	9887.242958	7100
2	11-15	10016.556955	1747
3	16-20	10019.469027	678
4	21-25	9486.944444	90
5	26-30	10342.391304	69

[893]:
```
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=grouped, x='cred_hist_group', y='avg_loan_amnt',␣
 ↪hue='cred_hist_group', palette='crest')

for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.3f}',
                (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom', fontsize=10)

plt.title('Average Loan Amount by Credit History Length')
plt.xlabel('Credit History Group')
plt.ylabel('Average Loan Amount')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[896]:
```
plt.figure(figsize=(8, 5))
sns.regplot(data=df,
            x='cb_person_cred_hist_length',
            y='loan_amnt',
            scatter_kws={'alpha': 0.3},
            line_kws={'color': 'red'})
plt.title('Relationship Between Loan Amount and Credit History Length')
plt.xlabel('Credit History Length (years)')
```

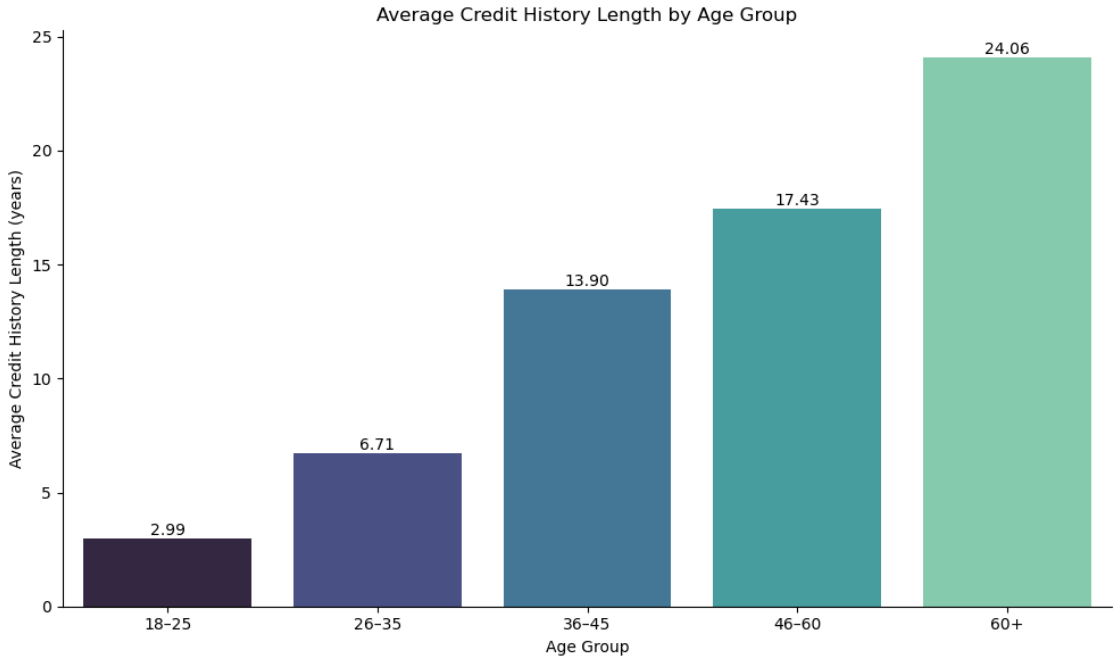
```
plt.ylabel('Loan Amount')
plt.tight_layout()
plt.show()
```



What is the average credit history length across age groups?

[898]:
```
age_credit_hist_length = df.groupby('age_group',␣
↪observed=True)['cb_person_cred_hist_length'].mean().reset_index()
```

[899]:
```
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=age_credit_hist_length, x='age_group',␣
↪y='cb_person_cred_hist_length', hue='age_group', palette='mako')

for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.2f}',
                (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom', fontsize=10)

plt.title('Average Credit History Length by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Average Credit History Length (years)')
sns.despine()
plt.tight_layout()
plt.show()
```



How is default history (cb_person_default_on_file) related to interest rate?

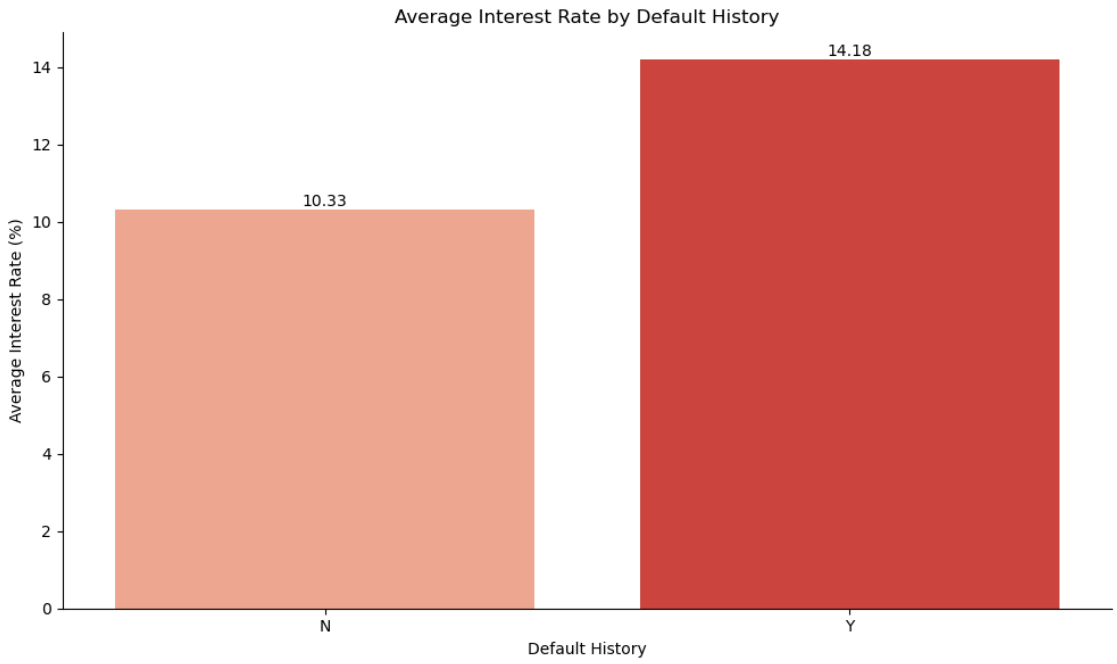
[904]:
```
df['cb_person_default_on_file'] = df['cb_person_default_on_file'].astype(str)

grouped = df.groupby('cb_person_default_on_file',␣
 ↪observed=True)['loan_int_rate'].agg(['mean', 'count']).reset_index()
grouped.rename(columns={'mean': 'avg_int_rate', 'count': 'clients'},␣
 ↪inplace=True)

grouped
```

[904]:

	cb_person_default_on_file	avg_int_rate	clients
0	N	10.327742	20079
1	Y	14.183341	4247

[905]:
```
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=grouped, x='cb_person_default_on_file', y='avg_int_rate',␣
 ↪hue='cb_person_default_on_file', palette='Reds')

for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.2f}',
                (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom', fontsize=10)

plt.title('Average Interest Rate by Default History')
```

```
plt.xlabel('Default History')
plt.ylabel('Average Interest Rate (%)')
sns.despine()
plt.tight_layout()
plt.show()
```

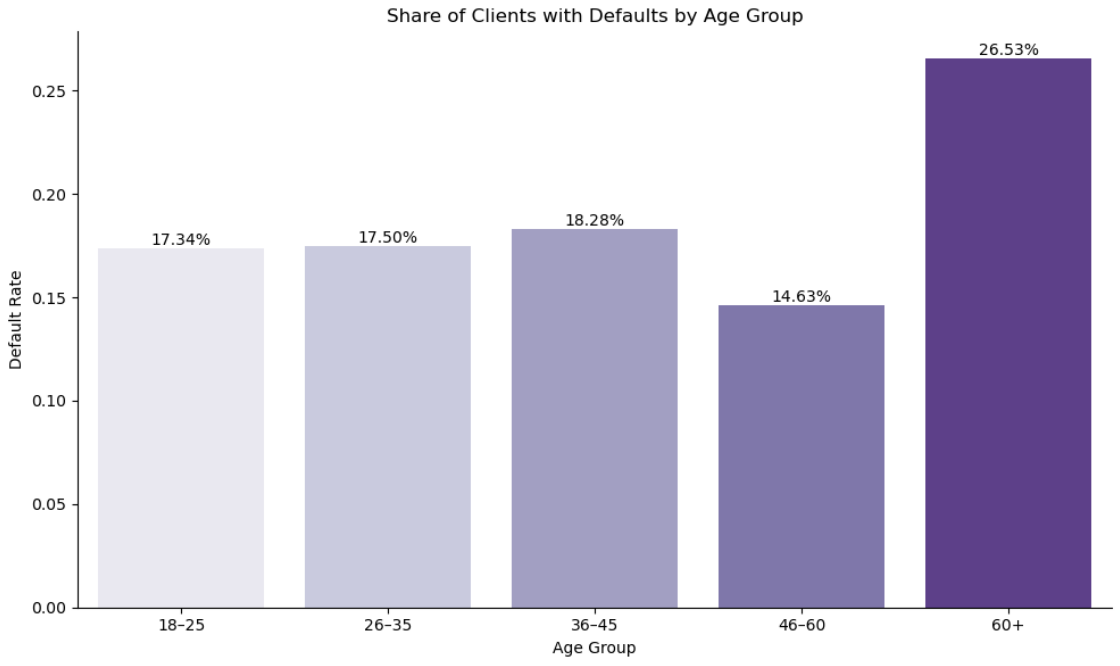


Who has more defaults — by age, loan purpose, or employment status?

[909]:
```
df['age_group'] = pd.cut(df['person_age'], bins=[17, 25, 35, 45, 60, 80],
                         labels=['18-25', '26-35', '36-45', '46-60', '60+'])
age_default = df.groupby('age_group',␣
 ↪observed=True)['cb_person_default_on_file'].apply(
    lambda x: (x == 'Y').mean()
).reset_index(name='default_rate')


intent_default = df.groupby('loan_intent',␣
 ↪observed=True)['cb_person_default_on_file'].apply(
    lambda x: (x == 'Y').mean()
).reset_index(name='default_rate')


df['emp_group'] = pd.cut(df['person_emp_length'], bins=[-1, 2, 5, 10, 20, 100],
                         labels=['0-2   ', '3-5   ', '6-10   ', '11-20   ', '20+␣
 ↪  '])
```

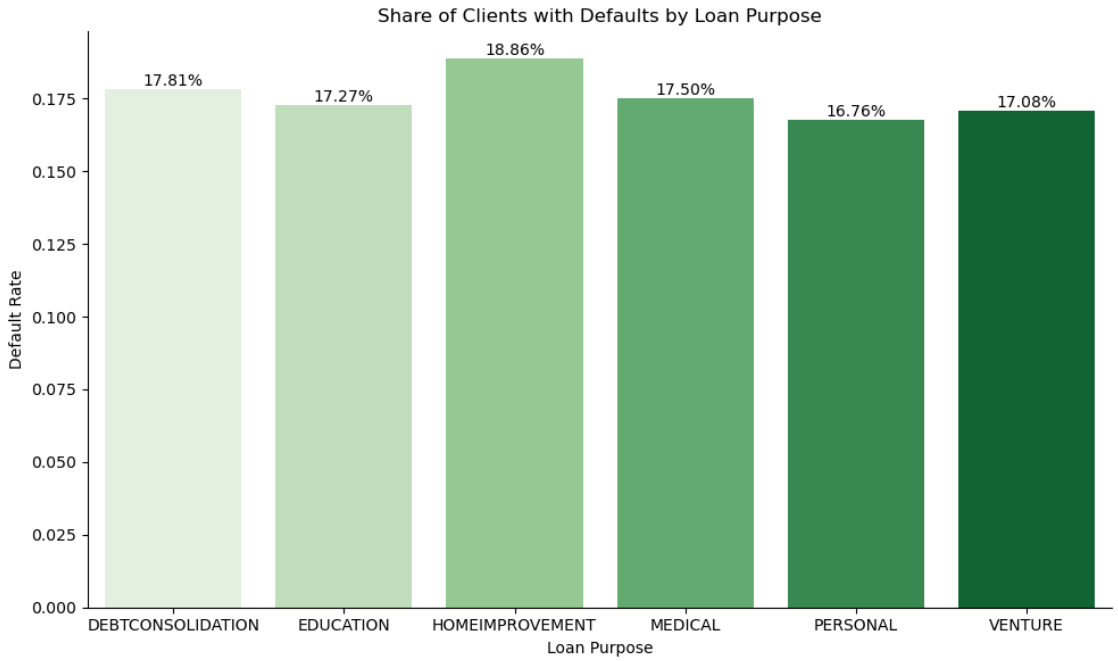
```
emp_default = df.groupby('emp_group',␣
 ↪observed=True)['cb_person_default_on_file'].apply(
    lambda x: (x == 'Y').mean()
).reset_index(name='default_rate')
```

[911]:
```
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=age_default, x='age_group', y='default_rate',␣
 ↪hue='age_group', palette='Purples')

for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.2%}',
                (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom', fontsize=10)

plt.title('Share of Clients with Defaults by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Default Rate')
sns.despine()
plt.tight_layout()
plt.show()
```



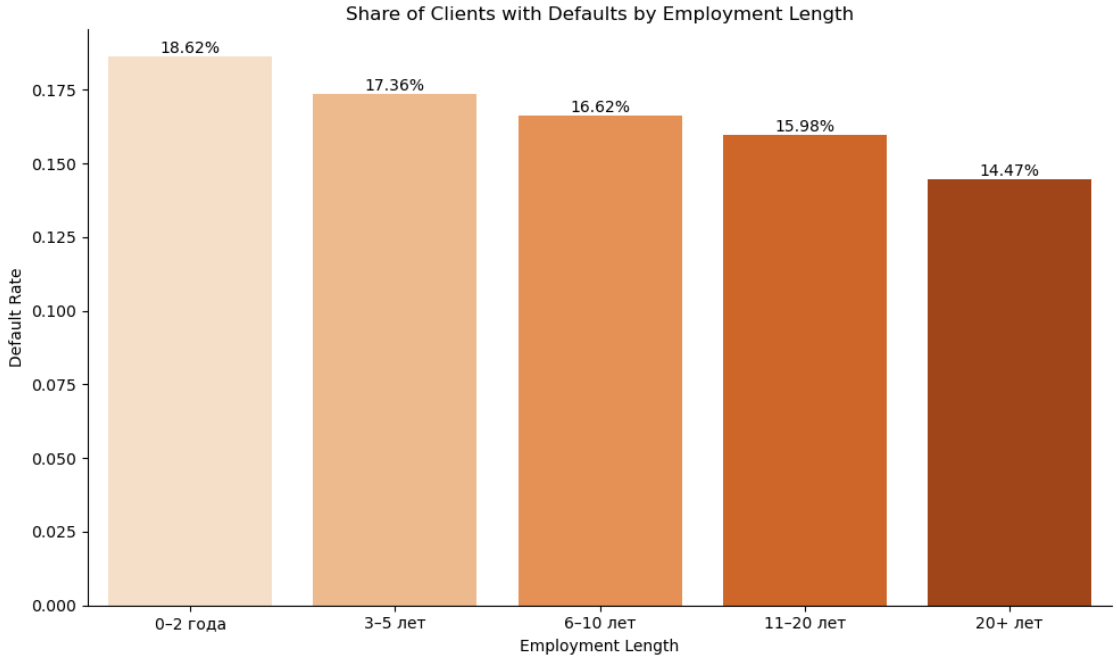
[913]:
```
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=intent_default, x='loan_intent', y='default_rate',␣
 ↪hue='loan_intent', palette='Greens')
```

```
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.2%}',
                (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom', fontsize=10)

plt.title('Share of Clients with Defaults by Loan Purpose')
plt.xlabel('Loan Purpose')
plt.ylabel('Default Rate')
sns.despine()
plt.tight_layout()
plt.show()
```



[915]:
```
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=emp_default, x='emp_group', y='default_rate',␣
 ↪hue='emp_group', palette='Oranges')

for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.2%}',
                (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom', fontsize=10)

plt.title('Share of Clients with Defaults by Employment Length')
```

```
plt.xlabel('Employment Length')
plt.ylabel('Default Rate')
sns.despine()
plt.tight_layout()
plt.show()
```



Dependence on Loan Purpose:

Which loan purposes are more frequently associated with larger loan amounts?

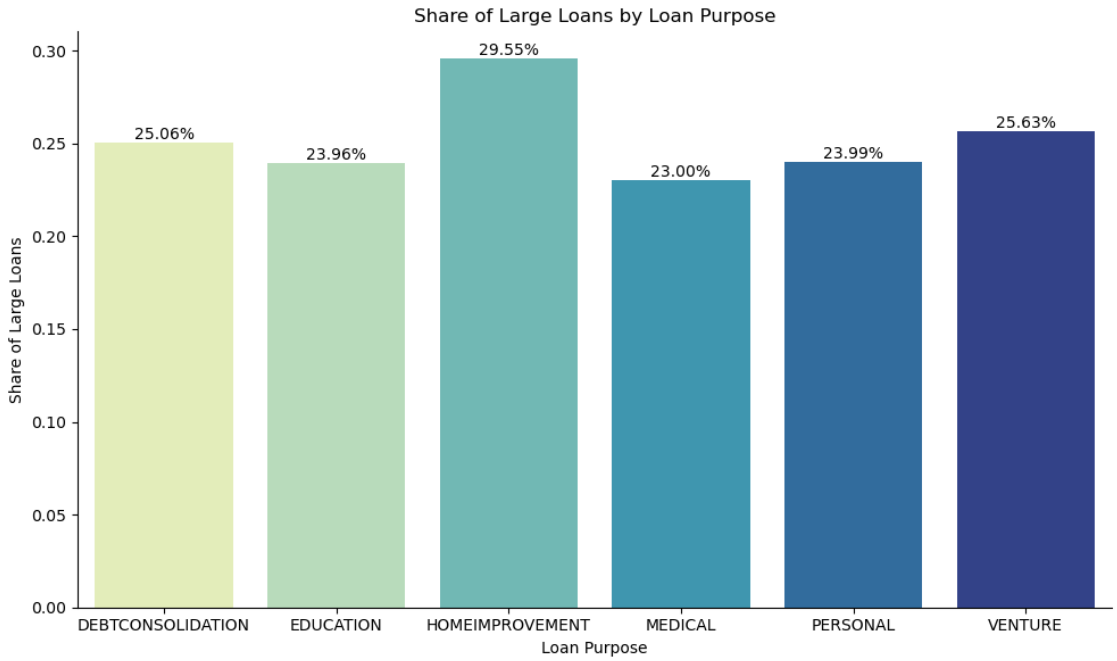
[918]:
```
large_loan_by_intent = df.groupby('loan_intent',␣
↪observed=True)['is_large_loan'].mean().reset_index()
large_loan_by_intent
```

[918]:

	loan_intent	is_large_loan
0	DEBTCONSOLIDATION	0.250632
1	EDUCATION	0.239557
2	HOMEIMPROVEMENT	0.295488
3	MEDICAL	0.229982
4	PERSONAL	0.239903
5	VENTURE	0.256302

[920]:
```
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=large_loan_by_intent, x='loan_intent', y='is_large_loan',␣
↪hue='loan_intent', palette='YlGnBu')
```

```
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.2%}',
                (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom', fontsize=10)

plt.title('Share of Large Loans by Loan Purpose')
plt.xlabel('Loan Purpose')
plt.ylabel('Share of Large Loans')
sns.despine()
plt.tight_layout()
plt.show()
```

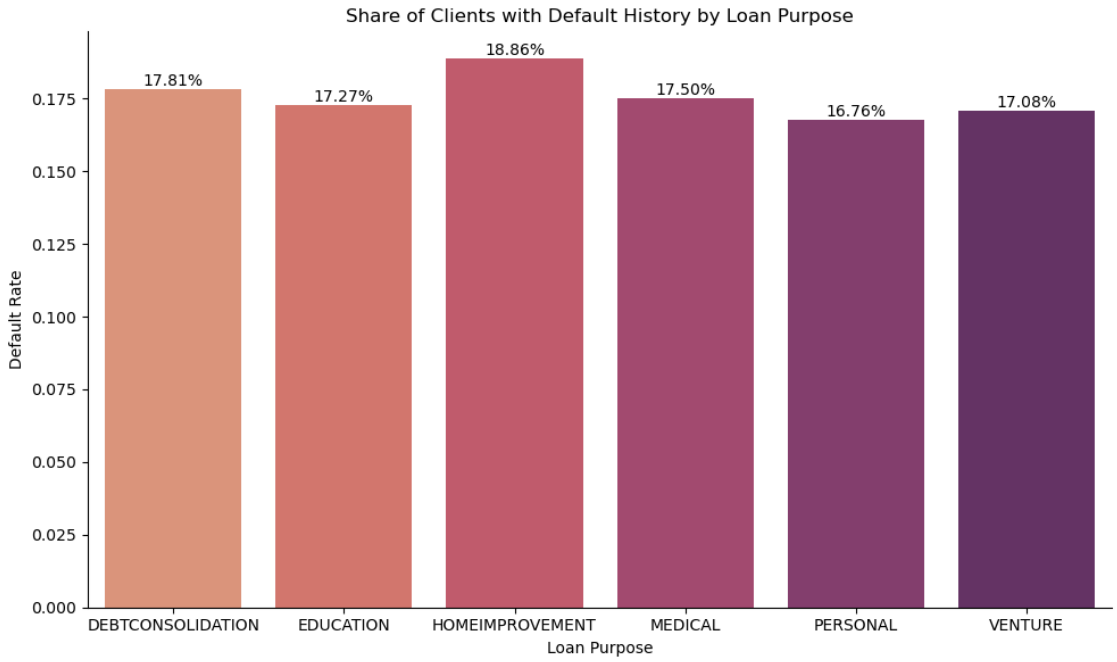


Which loan intents have a higher share of borrowers with default history?

[923]:
```
default_by_intent = df.groupby('loan_intent',␣
 ↪observed=True)['cb_person_default_on_file'].apply(
    lambda x: (x == 'Y').mean()
).reset_index(name='default_rate')
```

[925]:
```
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=default_by_intent, x='loan_intent', y='default_rate',␣
 ↪hue='loan_intent', palette='flare')

for p in ax.patches:
```

```
    height = p.get_height()
    ax.annotate(f'{height:.2%}',
                (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom', fontsize=10)

plt.title('Share of Clients with Default History by Loan Purpose')
plt.xlabel('Loan Purpose')
plt.ylabel('Default Rate')
sns.despine()
plt.tight_layout()
plt.show()
```



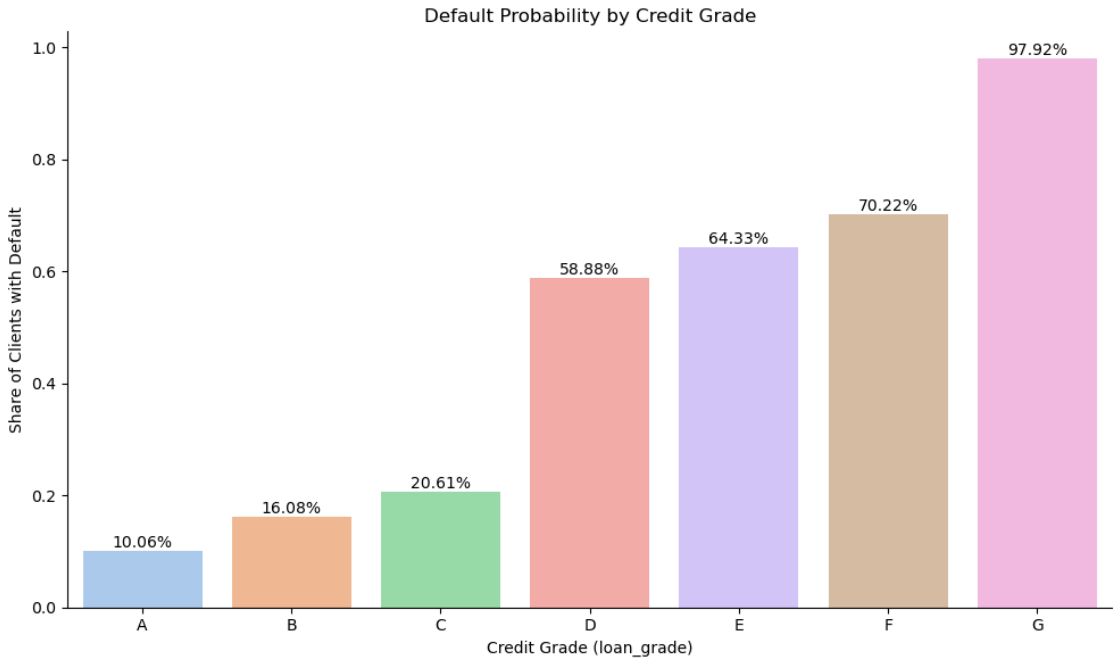
For the Classification Model:

How does loan_grade affect default probability?

[929]:
```
default_by_grade = df.groupby('loan_grade', observed=True)['loan_status'].
 ↪mean().reset_index()
```

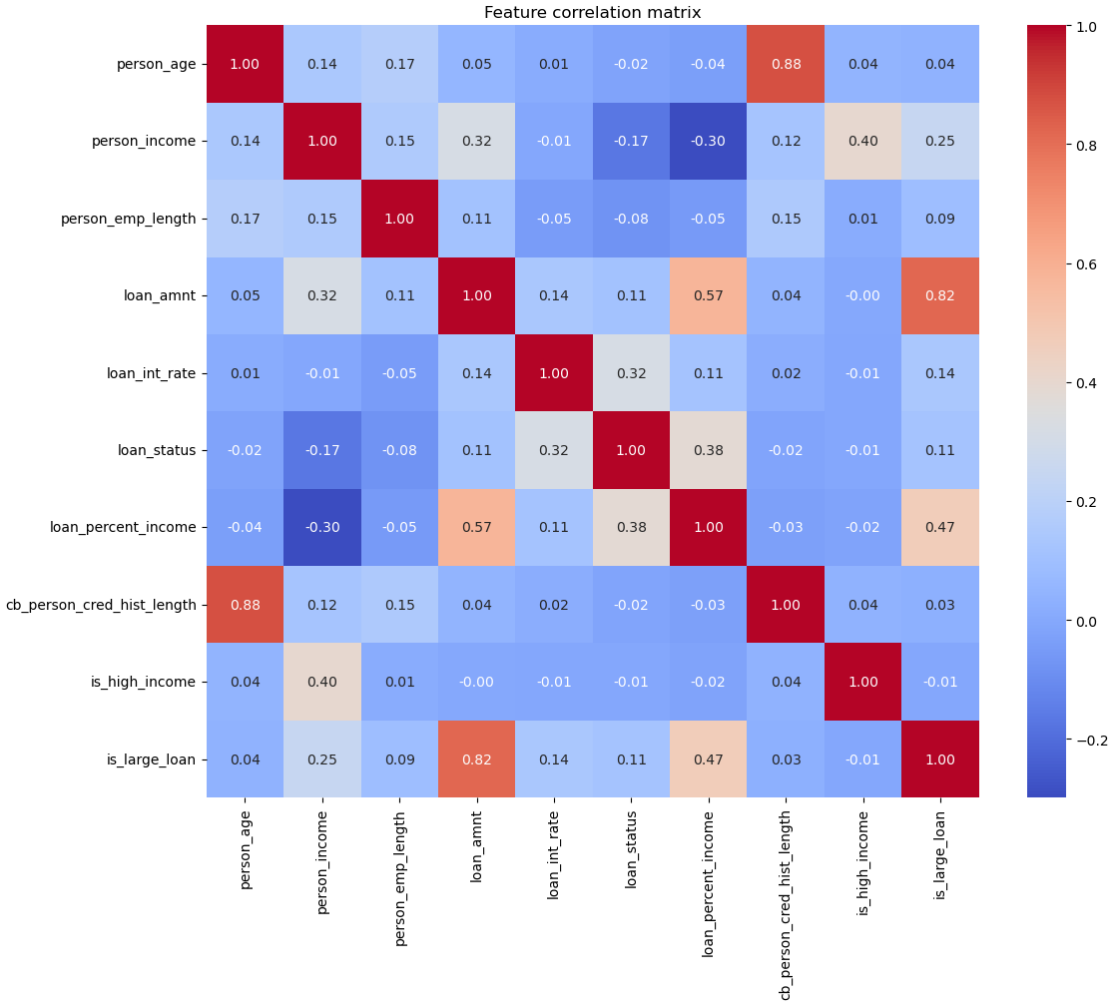
[931]:
```
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=default_by_grade, x='loan_grade', y='loan_status',␣
 ↪hue='loan_grade', palette='pastel')

for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height:.2%}',
```

```
                (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom', fontsize=10)

plt.title('Default Probability by Credit Grade')
plt.xlabel('Credit Grade (loan_grade)')
plt.ylabel('Share of Clients with Default')
sns.despine()
plt.tight_layout()
plt.show()
```



Which features are correlated?

[934]:
```
numeric_df = df.select_dtypes(include=['int64', 'float64'])

corr_matrix = numeric_df.corr()

plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm', square=True)
plt.title('Feature correlation matrix')
plt.tight_layout()
plt.show()
```



Encoding categorical variables and scaling all features

We encode categorical variables using the One-Hot Encoding method since these variables do not have a natural order.

[938]:
```
df = pd.get_dummies(df, columns=['person_home_ownership', 'loan_intent',␣
↪'loan_grade', 'cb_person_default_on_file'])
bool_columns = df.select_dtypes(include=['bool']).columns
df[bool_columns] = df[bool_columns].astype(int)
```

We normalize numerical variables to ensure consistency in scale. For variables with a small range, we use Min-Max Scaler. For other data, since they do not follow a normal distribution and contain outliers, we first apply Robust Scaler and then Min-Max Scaler to bring them to a common scale.

[941]:
```
df.describe()
```

[941]:

	person_age	person_income	person_emp_length	loan_amnt
count	24326.000000	2.432600e+04	24326.000000	24326.000000
mean	27.743525	6.593198e+04	4.755036	9589.160774
std	6.217010	5.190047e+04	3.986881	6303.199289
min	20.000000	4.000000e+03	0.000000	500.000000
25%	23.000000	3.900000e+04	2.000000	5000.000000
50%	26.000000	5.500000e+04	4.000000	8000.000000
75%	30.000000	7.943400e+04	7.000000	12250.000000
max	78.000000	1.900000e+06	41.000000	35000.000000

	loan_int_rate	loan_status	loan_percent_income
count	24326.000000	24326.000000	24326.000000
mean	11.000879	0.217052	0.170180
std	3.072789	0.412247	0.106772
min	5.420000	0.000000	0.000000
25%	8.490000	0.000000	0.090000
50%	10.990000	0.000000	0.150000
75%	13.110000	0.000000	0.230000
max	22.480000	1.000000	0.780000

	cb_person_cred_hist_length	is_high_income	is_large_loan	…
count	24326.000000	24326.000000	24326.000000	…
mean	5.816698	0.000206	0.248787	…
std	4.045444	0.014336	0.432319	…
min	2.000000	0.000000	0.000000	…
25%	3.000000	0.000000	0.000000	…
50%	4.000000	0.000000	0.000000	…
75%	8.000000	0.000000	0.000000	…
max	30.000000	1.000000	1.000000	…

	loan_intent_VENTURE	loan_grade_A	loan_grade_B	loan_grade_C
count	24326.000000	24326.000000	24326.000000	24326.000000
mean	0.174505	0.331538	0.321549	0.197073
std	0.379551	0.470776	0.467081	0.397796
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000
75%	0.000000	1.000000	1.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000

	loan_grade_D	loan_grade_E	loan_grade_F	loan_grade_G
count	24326.000000	24326.000000	24326.000000	24326.000000
mean	0.111856	0.028694	0.007317	0.001973
std	0.315195	0.166947	0.085229	0.044378
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000

```
75%         0.000000      0.000000      0.000000      0.000000
max         1.000000      1.000000      1.000000      1.000000

       cb_person_default_on_file_N  cb_person_default_on_file_Y
count                  24326.000000                 24326.000000
mean                       0.825413                     0.174587
std                        0.379621                     0.379621
min                        0.000000                     0.000000
25%                        1.000000                     0.000000
50%                        1.000000                     0.000000
75%                        1.000000                     0.000000
max                        1.000000                     1.000000

[8 rows x 29 columns]
```

[943]:
```
scaler = RobustScaler()
df['person_age'] = scaler.fit_transform(df[['person_age']])
df['person_income'] = scaler.fit_transform(df[['person_income']])
df['loan_amnt'] = scaler.fit_transform(df[['loan_amnt']])
df['person_emp_length'] = scaler.fit_transform(df[['person_emp_length']])
df['loan_int_rate'] = scaler.fit_transform(df[['loan_int_rate']])
df['cb_person_cred_hist_length'] = scaler.
 ↪fit_transform(df[['cb_person_cred_hist_length']])

scaler = MinMaxScaler()
df['person_age'] = scaler.fit_transform(df[['person_age']])
df['person_income'] = scaler.fit_transform(df[['person_income']])
df['loan_amnt'] = scaler.fit_transform(df[['loan_amnt']])
df['person_emp_length'] = scaler.fit_transform(df[['person_emp_length']])
df['loan_int_rate'] = scaler.fit_transform(df[['loan_int_rate']])
df['cb_person_cred_hist_length'] = scaler.
 ↪fit_transform(df[['cb_person_cred_hist_length']])
```

[945]:
```
df.columns
```

[945]:
```
Index(['person_age', 'person_income', 'person_emp_length', 'loan_amnt',
       'loan_int_rate', 'loan_status', 'loan_percent_income',
       'cb_person_cred_hist_length', 'is_high_income', 'income_group',
       'is_large_loan', 'age_group', 'work_group', 'cred_hist_group',
       'emp_group', 'person_home_ownership_MORTGAGE',
       'person_home_ownership_OTHER', 'person_home_ownership_OWN',
       'person_home_ownership_RENT', 'loan_intent_DEBTCONSOLIDATION',
       'loan_intent_EDUCATION', 'loan_intent_HOMEIMPROVEMENT',
       'loan_intent_MEDICAL', 'loan_intent_PERSONAL', 'loan_intent_VENTURE',
       'loan_grade_A', 'loan_grade_B', 'loan_grade_C', 'loan_grade_D',
       'loan_grade_E', 'loan_grade_F', 'loan_grade_G',
       'cb_person_default_on_file_N', 'cb_person_default_on_file_Y'],
```

```
      dtype='object')
```

[947]:
```
df = df.drop(columns=['age_group', 'work_group', 'cred_hist_group',␣
 ↪'emp_group', 'income_group'])
df
```

[947]:

	person_age	person_income	person_emp_length	loan_amnt	loan_int_rate	\
0	0.103448	0.103428	0.243902	0.855072	0.368699	
1	0.189655	0.031883	0.073171	0.333333	0.428488	
2	0.034483	0.009648	0.121951	0.086957	0.648886	
3	0.224138	0.055380	0.097561	0.420290	0.355803	
4	0.068966	0.050633	0.195122	0.472464	0.144197	
…	…	…	…	…	…	
24430	0.327586	0.018196	0.170732	0.086957	0.501758	
24431	0.086207	0.034283	0.121951	0.231884	0.087925	
24432	0.103448	0.075949	0.024390	0.565217	0.736811	
24433	0.103448	0.090190	0.000000	0.420290	0.326495	
24434	0.120690	0.032173	0.268293	0.217391	0.121336	

	loan_status	loan_percent_income	cb_person_cred_hist_length	\
0	0	0.15	0.071429	
1	0	0.19	0.214286	
2	1	0.16	0.035714	
3	0	0.14	0.107143	
4	0	0.17	0.071429	
…	…	…	…	
24430	0	0.09	0.535714	
24431	1	0.12	0.071429	
24432	1	0.14	0.035714	
24433	0	0.09	0.035714	
24434	0	0.12	0.214286	

	is_high_income	is_large_loan	…	loan_intent_VENTURE	loan_grade_A	\
0	0	1	…	0	0	
1	0	0	…	1	0	
2	0	0	…	0	0	
3	0	1	…	0	0	
4	0	1	…	0	1	
…	…	…	…	…	…	
24430	0	0	…	0	0	
24431	0	0	…	0	1	
24432	0	1	…	0	0	
24433	0	1	…	0	0	
24434	0	0	…	1	1	

	loan_grade_B	loan_grade_C	loan_grade_D	loan_grade_E	loan_grade_F	\
0	1	0	0	0	0	

```
1                0              1              0              0              0
2                0              0              1              0              0
3                1              0              0              0              0
4                0              0              0              0              0
…              …              …              …              …              …
24430            0              1              0              0              0
24431            0              0              0              0              0
24432            0              0              0              1              0
24433            0              1              0              0              0
24434            0              0              0              0              0

       loan_grade_G  cb_person_default_on_file_N  cb_person_default_on_file_Y
0                 0                            1                            0
1                 0                            1                            0
2                 0                            1                            0
3                 0                            1                            0
4                 0                            1                            0
…               …                            …                            …
24430             0                            0                            1
24431             0                            1                            0
24432             0                            1                            0
24433             0                            1                            0
24434             0                            1                            0

[24326 rows x 29 columns]
```

[949]:
```
df.describe()
```

[949]:
```
         person_age  person_income  person_emp_length     loan_amnt  \
count  24326.000000   24326.000000       24326.000000  24326.000000
mean       0.133509       0.032665           0.115976      0.263454
std        0.107190       0.027374           0.097241      0.182701
min        0.000000       0.000000           0.000000      0.000000
25%        0.051724       0.018460           0.048780      0.130435
50%        0.103448       0.026899           0.097561      0.217391
75%        0.172414       0.039786           0.170732      0.340580
max        1.000000       1.000000           1.000000      1.000000

       loan_int_rate   loan_status  loan_percent_income  \
count   24326.000000  24326.000000         24326.000000
mean        0.327132      0.217052             0.170180
std         0.180117      0.412247             0.106772
min         0.000000      0.000000             0.000000
25%         0.179953      0.000000             0.090000
50%         0.326495      0.000000             0.150000
75%         0.450762      0.000000             0.230000
max         1.000000      1.000000             0.780000
```

```
       cb_person_cred_hist_length  is_high_income  is_large_loan  …  \
count                24326.000000    24326.000000   24326.000000  …
mean                     0.136311        0.000206       0.248787  …
std                      0.144480        0.014336       0.432319  …
min                      0.000000        0.000000       0.000000  …
25%                      0.035714        0.000000       0.000000  …
50%                      0.071429        0.000000       0.000000  …
75%                      0.214286        0.000000       0.000000  …
max                      1.000000        1.000000       1.000000  …

       loan_intent_VENTURE  loan_grade_A  loan_grade_B  loan_grade_C  \
count         24326.000000  24326.000000  24326.000000  24326.000000
mean              0.174505      0.331538      0.321549      0.197073
std               0.379551      0.470776      0.467081      0.397796
min               0.000000      0.000000      0.000000      0.000000
25%               0.000000      0.000000      0.000000      0.000000
50%               0.000000      0.000000      0.000000      0.000000
75%               0.000000      1.000000      1.000000      0.000000
max               1.000000      1.000000      1.000000      1.000000

       loan_grade_D  loan_grade_E  loan_grade_F  loan_grade_G  \
count  24326.000000  24326.000000  24326.000000  24326.000000
mean       0.111856      0.028694      0.007317      0.001973
std        0.315195      0.166947      0.085229      0.044378
min        0.000000      0.000000      0.000000      0.000000
25%        0.000000      0.000000      0.000000      0.000000
50%        0.000000      0.000000      0.000000      0.000000
75%        0.000000      0.000000      0.000000      0.000000
max        1.000000      1.000000      1.000000      1.000000

       cb_person_default_on_file_N  cb_person_default_on_file_Y
count                 24326.000000                 24326.000000
mean                      0.825413                     0.174587
std                       0.379621                     0.379621
min                       0.000000                     0.000000
25%                       1.000000                     0.000000
50%                       1.000000                     0.000000
75%                       1.000000                     0.000000
max                       1.000000                     1.000000

[8 rows x 29 columns]
```

Model building

Which features are most informative for predicting default?

[953]:
```
X = df.drop(columns=['loan_status'])
y = df['loan_status']

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,␣
 ↪test_size=0.2, random_state=42)

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

y_pred_proba = model.predict_proba(X_test)[:, 1]

print(f"AUC: {roc_auc_score(y_test, y_pred_proba):.4f}")
print(classification_report(y_test, model.predict(X_test)))
```

```
AUC: 0.9269
              precision    recall  f1-score   support

           0       0.93      0.99      0.96      3810
           1       0.96      0.72      0.82      1056

    accuracy                           0.93      4866
   macro avg       0.94      0.86      0.89      4866
weighted avg       0.93      0.93      0.93      4866
```

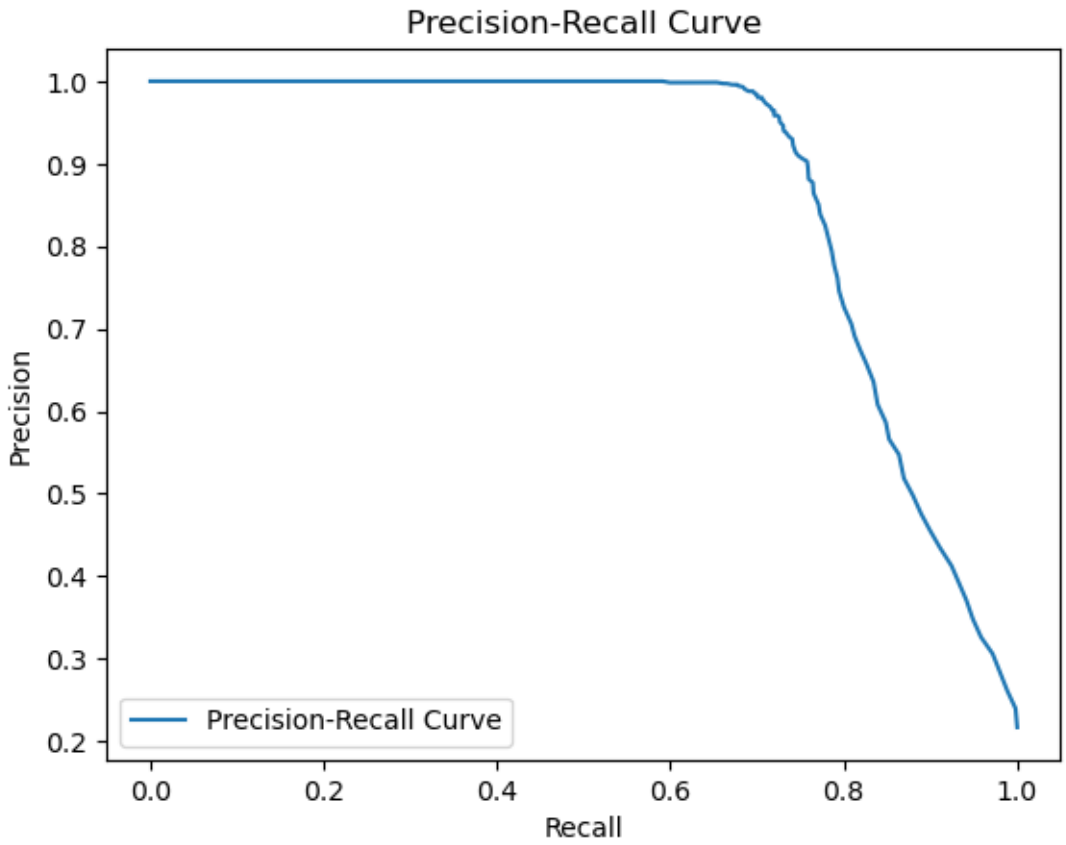
[954]:
```
rf = RandomForestClassifier(random_state=42)

param_dist = {
    'n_estimators': randint(100, 300),
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2']
}

random_search = RandomizedSearchCV(
    estimator=rf,
    param_distributions=param_dist,
    n_iter=30,
    cv=5,
    scoring='roc_auc',
    n_jobs=-1,
    verbose=2,
    random_state=42
)
random_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 30 candidates, totalling 150 fits
```

[954]:
```
RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
                   n_iter=30, n_jobs=-1,
                   param_distributions={'max_depth': [10, 20, None],
                                        'max_features': ['sqrt', 'log2'],
                                        'min_samples_leaf': [1, 2, 4],
                                        'min_samples_split': [2, 5, 10],
                                        'n_estimators':
 <scipy.stats._distn_infrastructure.rv_discrete_frozen object at 0x30101ede0>},
                   random_state=42, scoring='roc_auc', verbose=2)
```

[955]:
```
precision, recall, thresholds = precision_recall_curve(y_test, y_pred_proba)

f1_scores = 2 * (precision * recall) / (precision + recall)
optimal_threshold = thresholds[f1_scores.argmax()]

print("Optimal threshold:", optimal_threshold)

plt.plot(recall, precision, label="Precision-Recall Curve")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve")
plt.legend()
plt.show()
```

```
Optimal threshold: 0.49
```



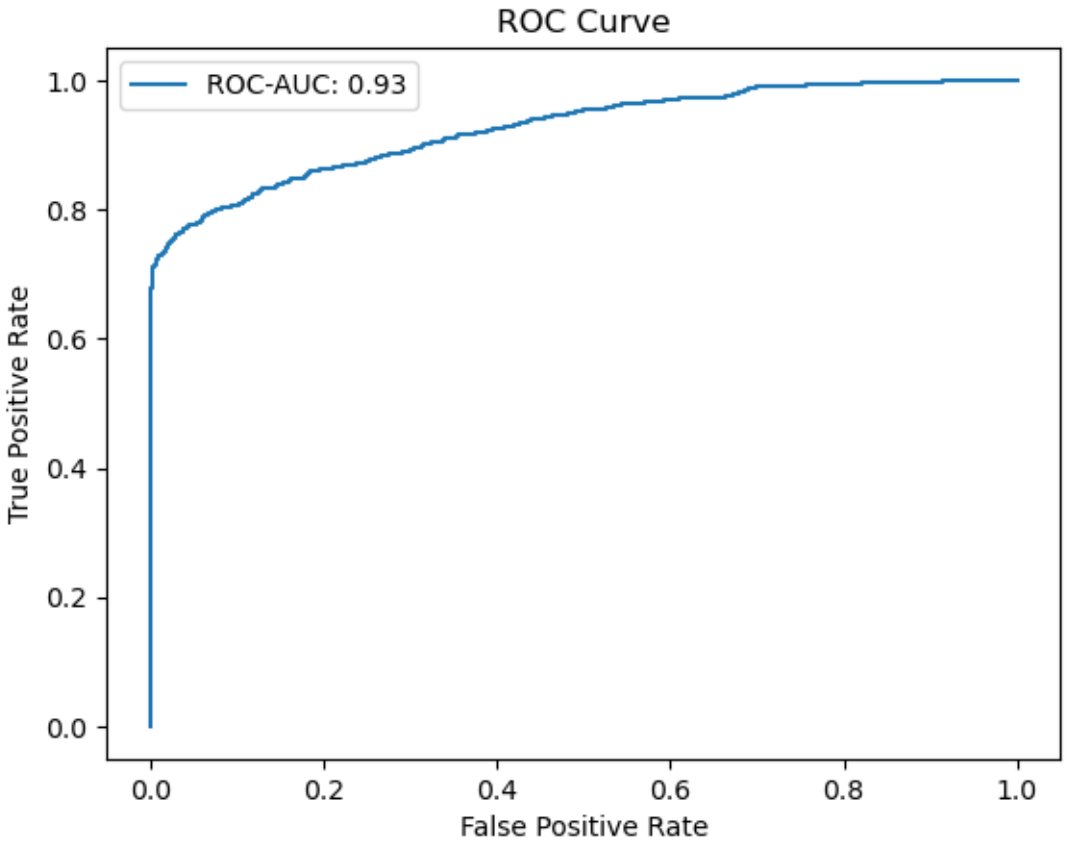
[956]:
```
X = df.drop(columns=['loan_status'])
y = df['loan_status']

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,␣
 ↪test_size=0.2, random_state=42)

model = RandomForestClassifier(max_depth=20, min_samples_leaf=2,␣
 ↪min_samples_split=5,
                        n_estimators=233, random_state=42)
model.fit(X_train, y_train)

y_pred_proba = model.predict_proba(X_test)[:, 1]
y_pred = (y_pred_proba >= optimal_threshold).astype(int)

print(f"AUC: {roc_auc_score(y_test, y_pred_proba):.4f}")
print(classification_report(y_test, y_pred))
```

```
AUC: 0.9262
              precision    recall  f1-score   support
```

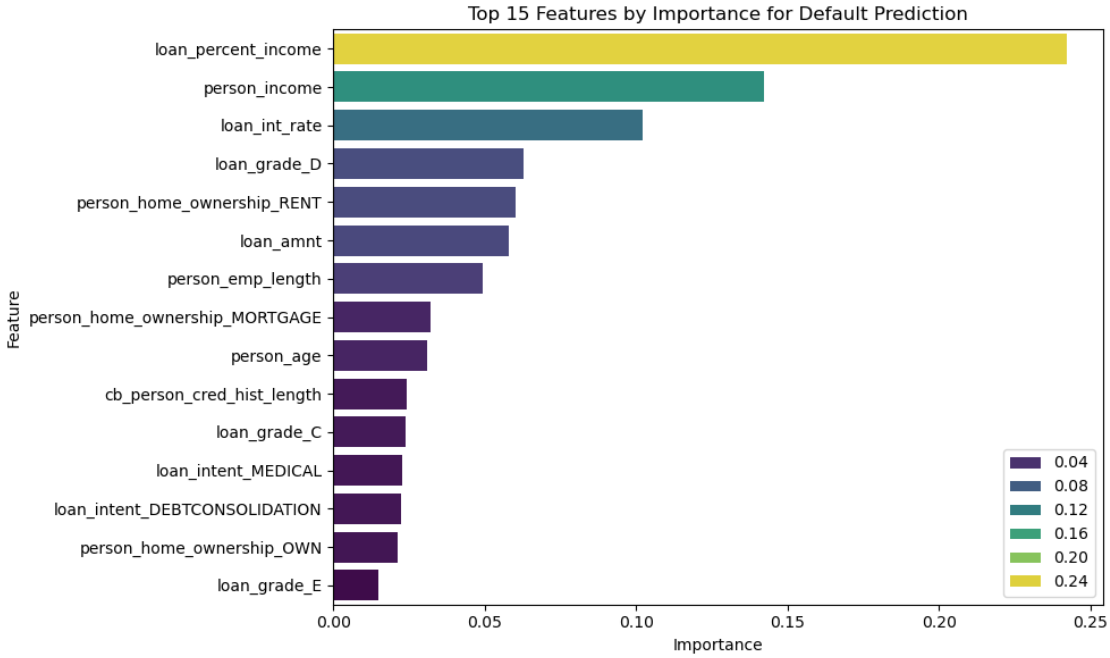
```
              0       0.93      0.99      0.96      3810
              1       0.97      0.72      0.82      1056

    accuracy                            0.93      4866
   macro avg        0.95      0.86      0.89      4866
weighted avg        0.94      0.93      0.93      4866
```

[957]:
```
roc_auc = roc_auc_score(y_test, y_pred_proba)
print(f"ROC-AUC Score: {roc_auc}")

fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
plt.plot(fpr, tpr, label=f"ROC-AUC: {roc_auc:.2f}")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()
```

```
ROC-AUC Score: 0.926211176727909
```



[959]:
```
importances = pd.Series(model.feature_importances_, index=X.columns)
top_features = importances.sort_values(ascending=False).head(15)

plt.figure(figsize=(10, 6))
sns.barplot(x=top_features.values, y=top_features.index, hue=top_features.
 ↪values, palette='viridis')
plt.title('Top 15 Features by Importance for Default Prediction')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()
```



[960]:
```
cv_auc = cross_val_score(model, X, y, cv=5, scoring='roc_auc')
print(f"Average CV AUC: {cv_auc.mean():.4f} ± {cv_auc.std():.4f}")
```

```
Average CV AUC: 0.9278 ± 0.0083
```

[961]:
```
new_data = pd.read_csv('credit_risk_new_data.csv')
```

[962]:
```
new_data = new_data.drop(columns=['loan_status'])
```

[963]:
```
new_data
```

[963]:

	person_age	person_income	person_home_ownership	person_emp_length	\
0	24	28000	OWN	6.0	
1	27	64000	RENT	0.0	

```
2              26          72000              MORTGAGE               10.0
3              23          27996                  RENT                7.0
4              30          44500                  RENT                2.0
…             …             …                     …                   …
8141           38          66000                  RENT                1.0
8142           27          39000                  RENT                3.0
8143           26          70000                  RENT                5.0
8144           24          10980                   OWN                0.0
8145           26          54600              MORTGAGE               10.0

            loan_intent loan_grade  loan_amnt  loan_int_rate  \
0       HOMEIMPROVEMENT          B      10000          10.37
1              PERSONAL          C      10000          15.27
2             EDUCATION          D      16000            NaN
3     DEBTCONSOLIDATION          A      10000            NaN
4               MEDICAL          E      13000          16.32
…                  …         …         …             …
8141           PERSONAL          D      10000          15.99
8142           PERSONAL          B      11000            NaN
8143           PERSONAL          C       4000          13.85
8144           PERSONAL          A       1500           7.29
8145           PERSONAL          B       4800            NaN

      loan_percent_income cb_person_default_on_file  \
0                    0.36                         N
1                    0.16                         Y
2                    0.22                         N
3                    0.36                         N
4                    0.29                         N
…                   …                            …
8141                 0.15                         Y
8142                 0.28                         N
8143                 0.06                         N
8144                 0.14                         N
8145                 0.09                         N

      cb_person_cred_hist_length
0                              2
1                             10
2                              3
3                              2
4                              6
…                             …
8141                          12
8142                           7
8143                           4
8144                           3
```

```
8145                          2

[8146 rows x 11 columns]
```

Lets prepare new data

[965]:
```
new_data.duplicated().sum()
```

[965]:
```
9
```

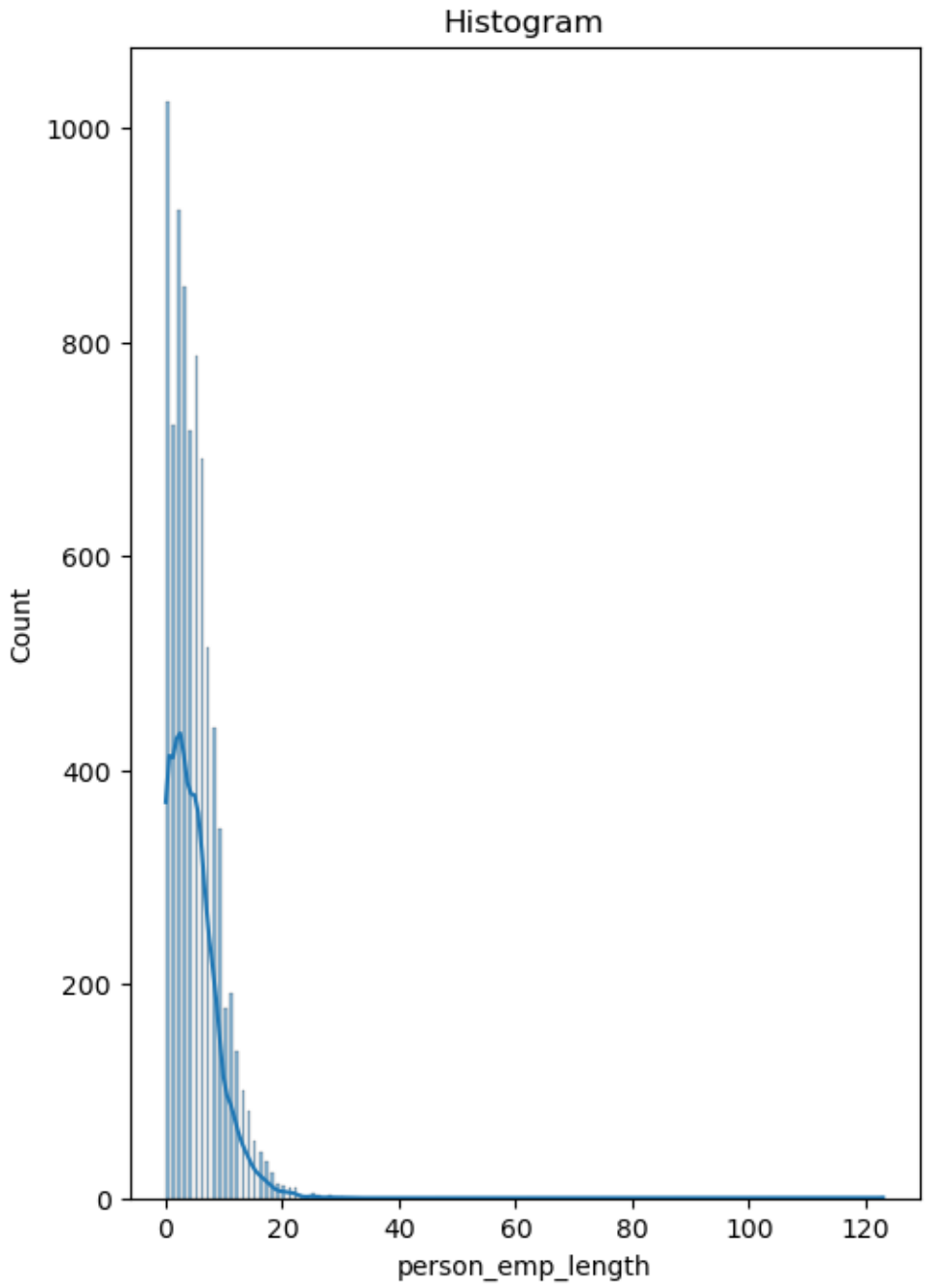
[966]:
```
new_data = new_data.drop_duplicates()
new_data.duplicated().sum()
```

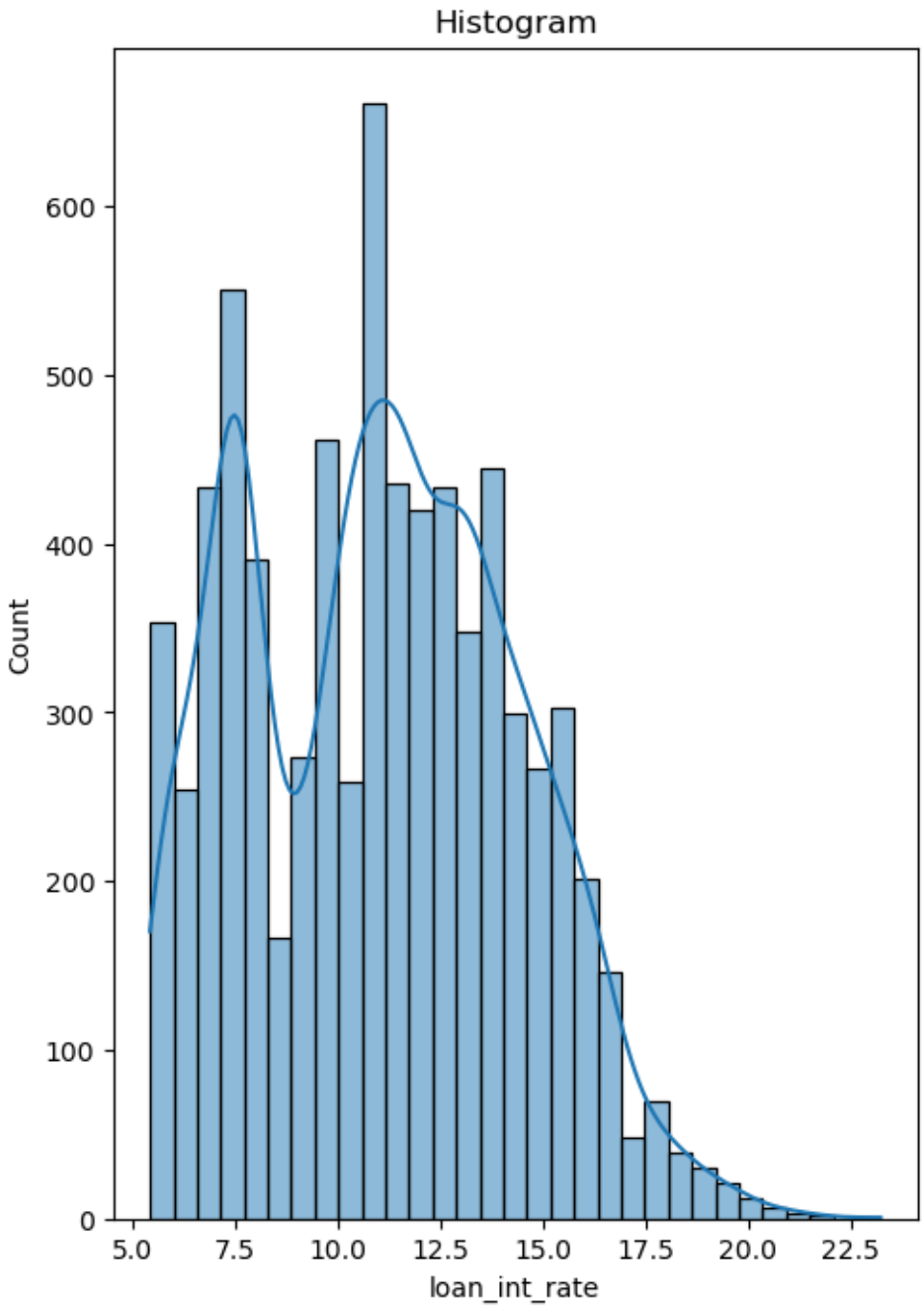
[966]:
```
0
```

[967]:
```
new_data.isnull().sum()
```

[967]:
```
person_age                    0
person_income                 0
person_home_ownership         0
person_emp_length           217
loan_intent                   0
loan_grade                    0
loan_amnt                     0
loan_int_rate               807
loan_percent_income           0
cb_person_default_on_file     0
cb_person_cred_hist_length    0
dtype: int64
```

[968]:
```
columns = ['person_emp_length', 'loan_int_rate']

for column in columns:
    plt.figure(figsize=(12, 8))
    plt.subplot(1, 2, 1)
    sns.histplot(new_data[column].dropna(), kde=True)
    plt.title('Histogram')
    plt.show()
```


Histogram
Count
person_emp_length
0
200
400
600
800
1000
0
20
40
60
80
100
120



[969]:
```
for column in columns:
    new_data[column] = new_data[column].fillna(new_data[column].median())
```

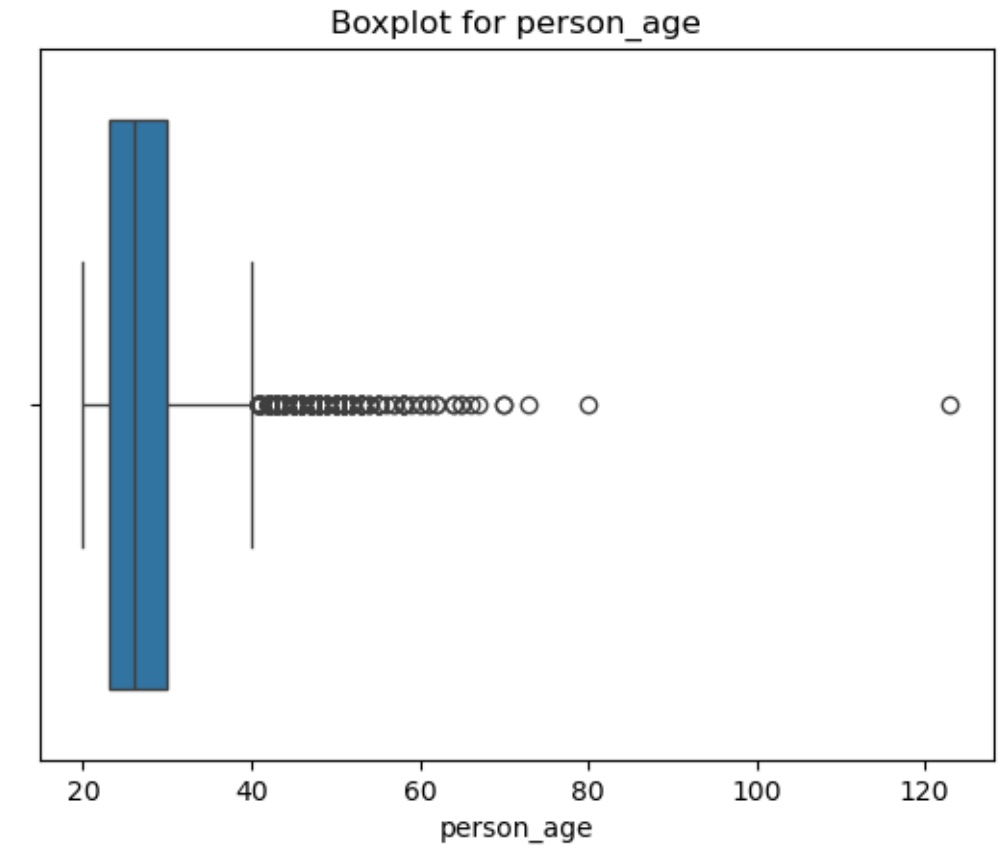
```
/var/folders/g_/dlksrxdd3pz88bqsmz91cx540000gn/T/ipykernel_28061/2743691635.py:2
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

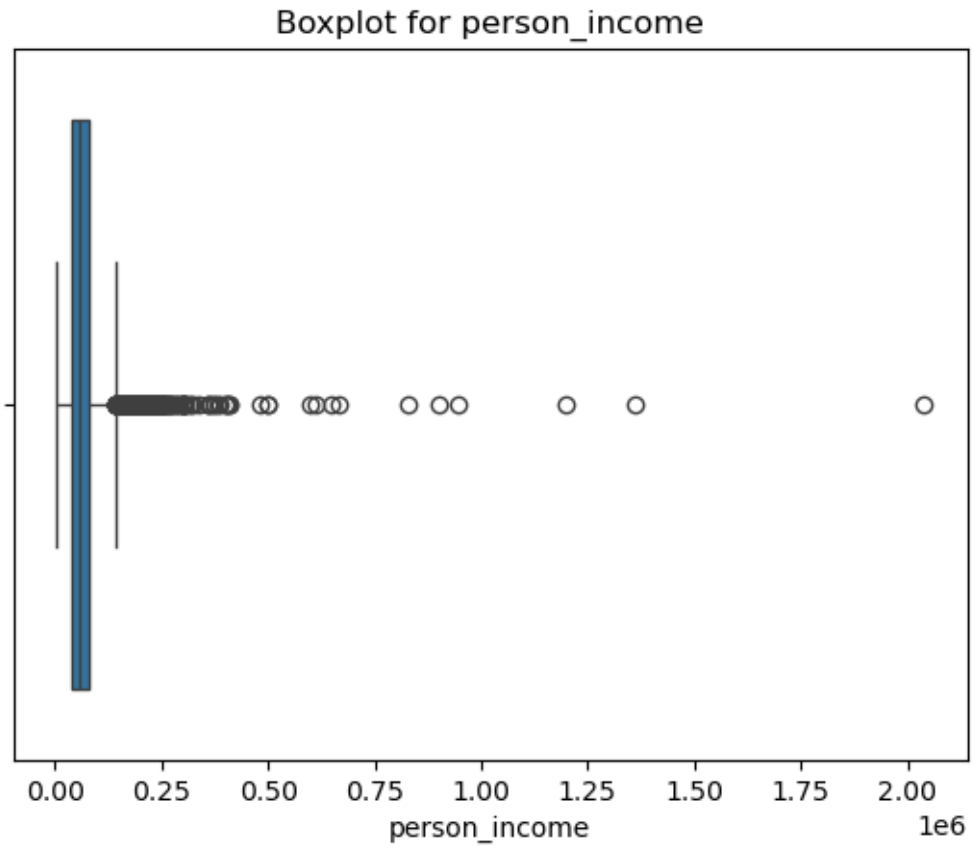
```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_data[column] = new_data[column].fillna(new_data[column].median())
```

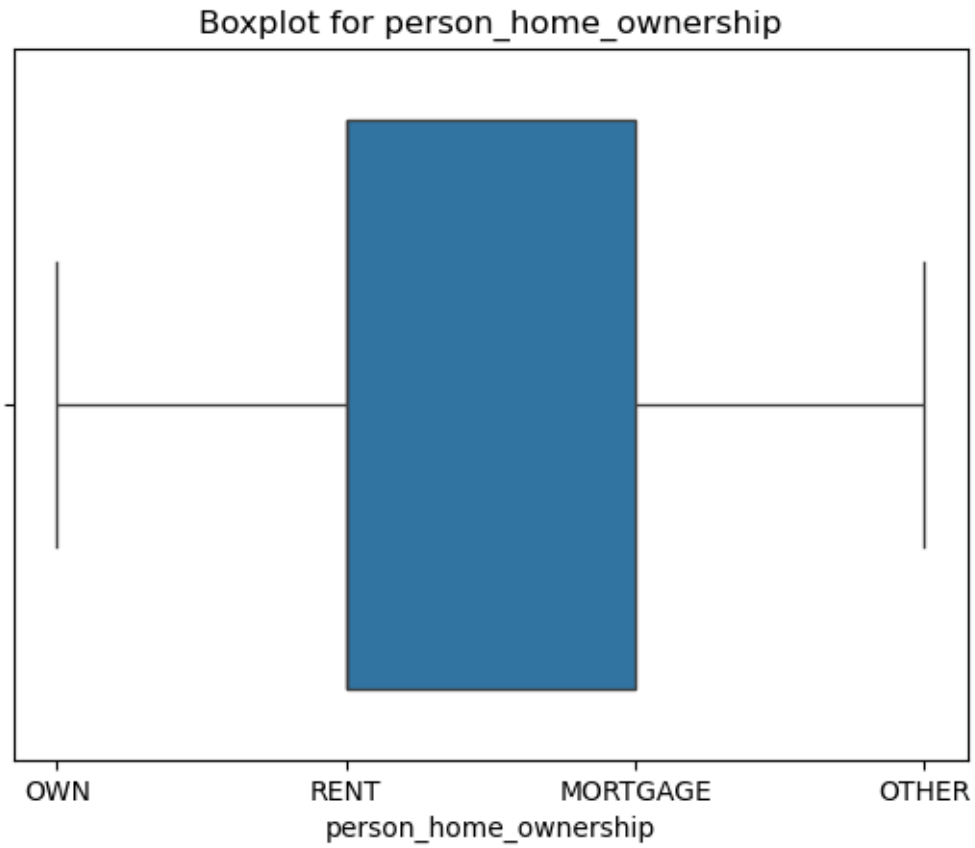
[970]:
```
new_data.isnull().sum()
```

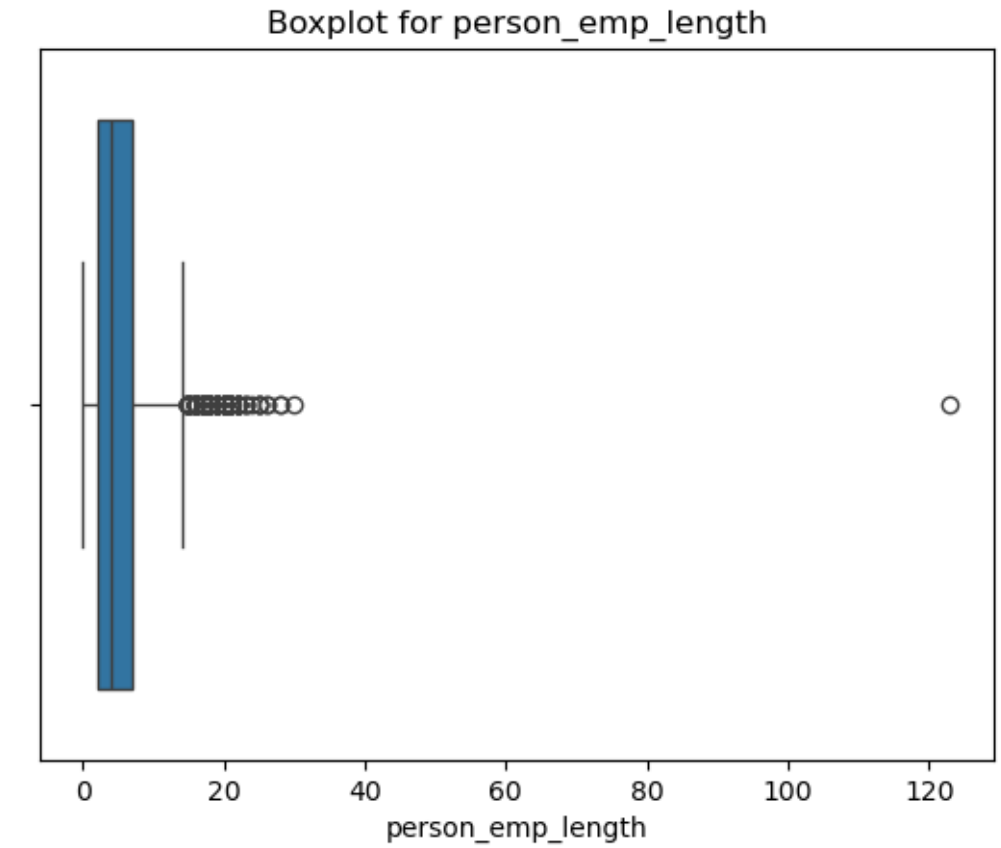
[970]:
```
person_age                  0
person_income               0
person_home_ownership       0
person_emp_length           0
loan_intent                 0
loan_grade                  0
loan_amnt                   0
loan_int_rate               0
loan_percent_income         0
cb_person_default_on_file   0
cb_person_cred_hist_length  0
dtype: int64
```

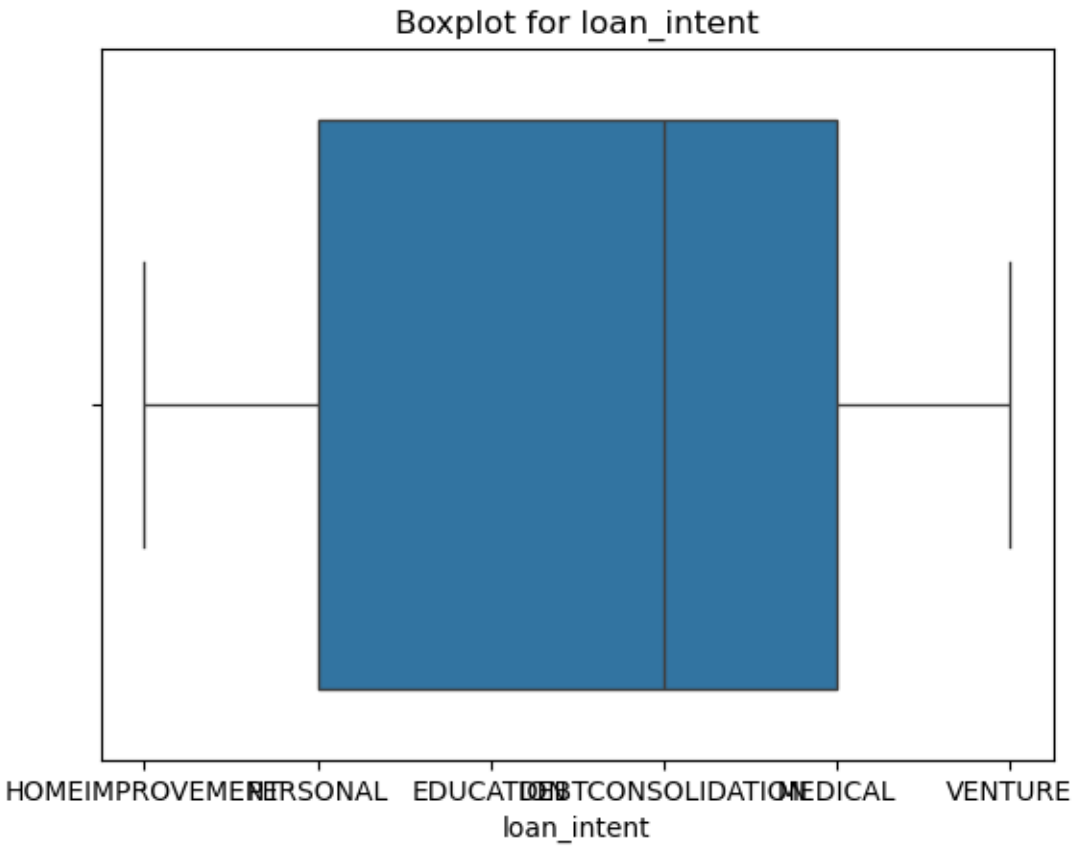
[971]:
```
for column in new_data.columns:
    sns.boxplot(data=new_data, x=column)
    plt.title(f'Boxplot for {column}')
    plt.show()
```

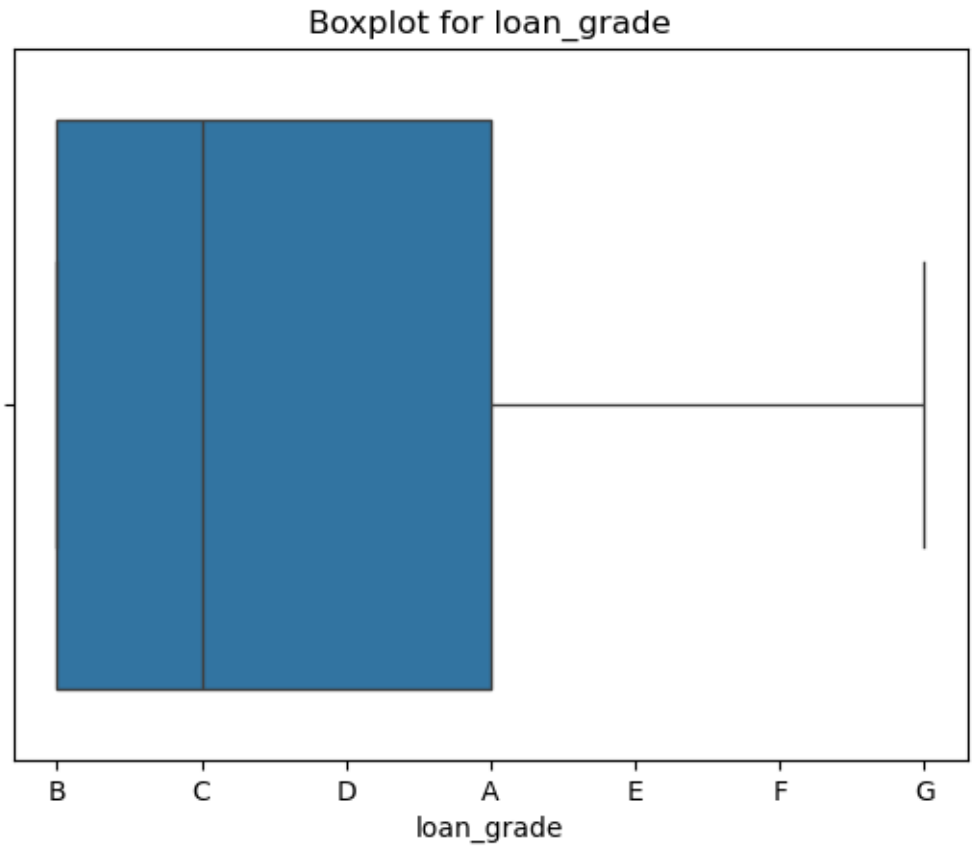

Boxplot for person_age
20
40
60
80
100
120
person_age

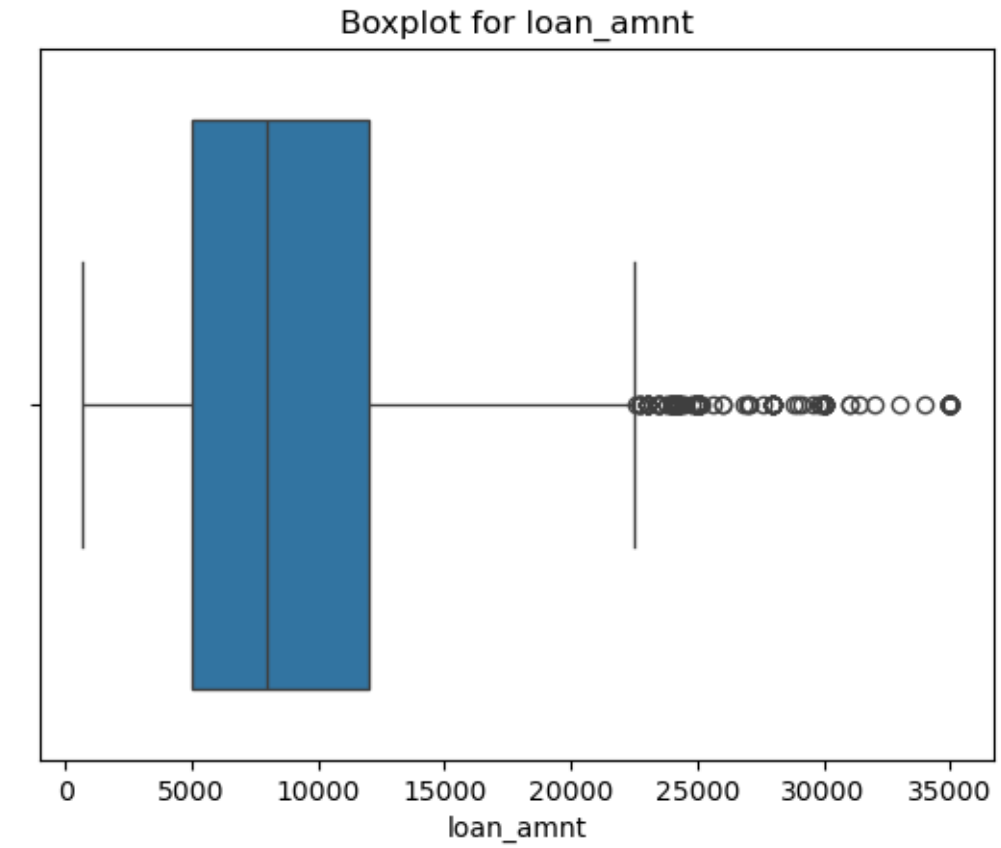

Boxplot for person_income
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
person_income
1e6

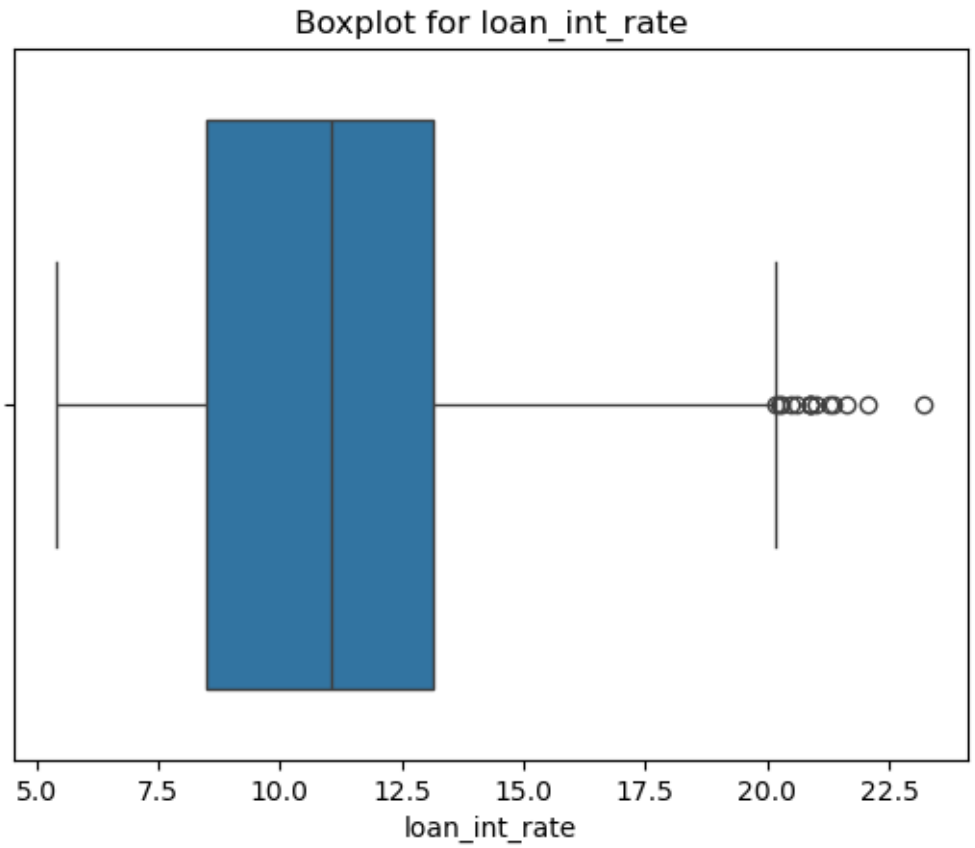

Boxplot for person_home_ownership
OWN
RENT
MORTGAGE
OTHER
person_home_ownership

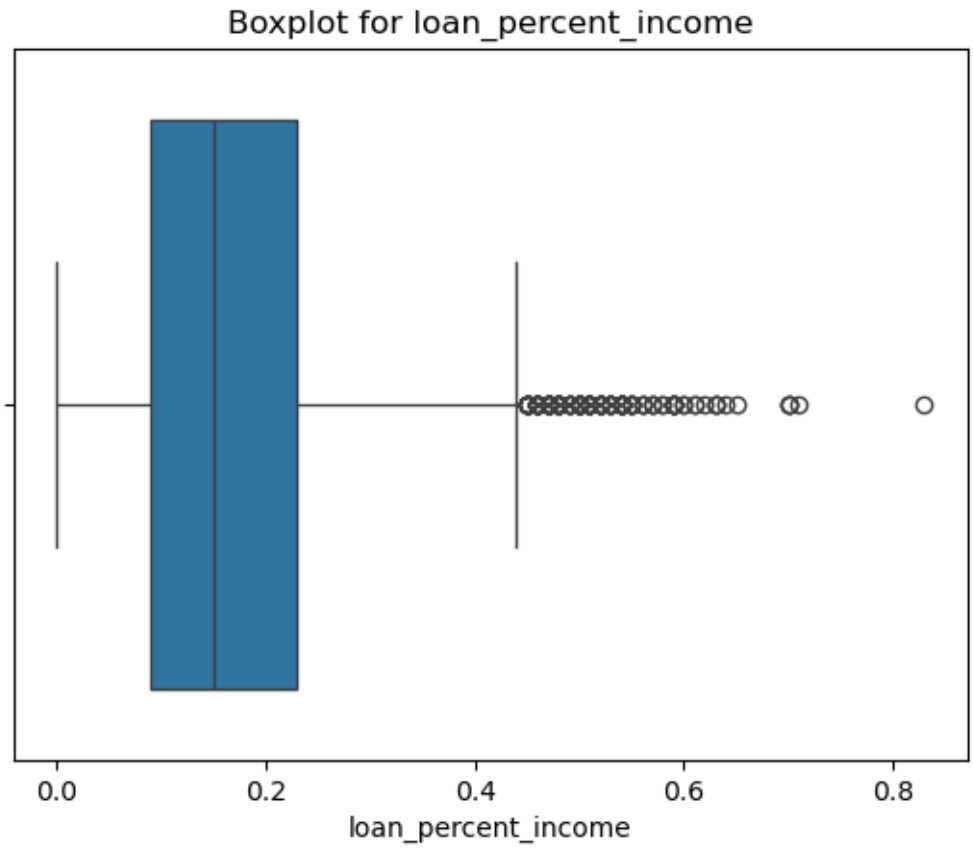

Boxplot for person_emp_length
0
20
40
60
80
100
120
person_emp_length

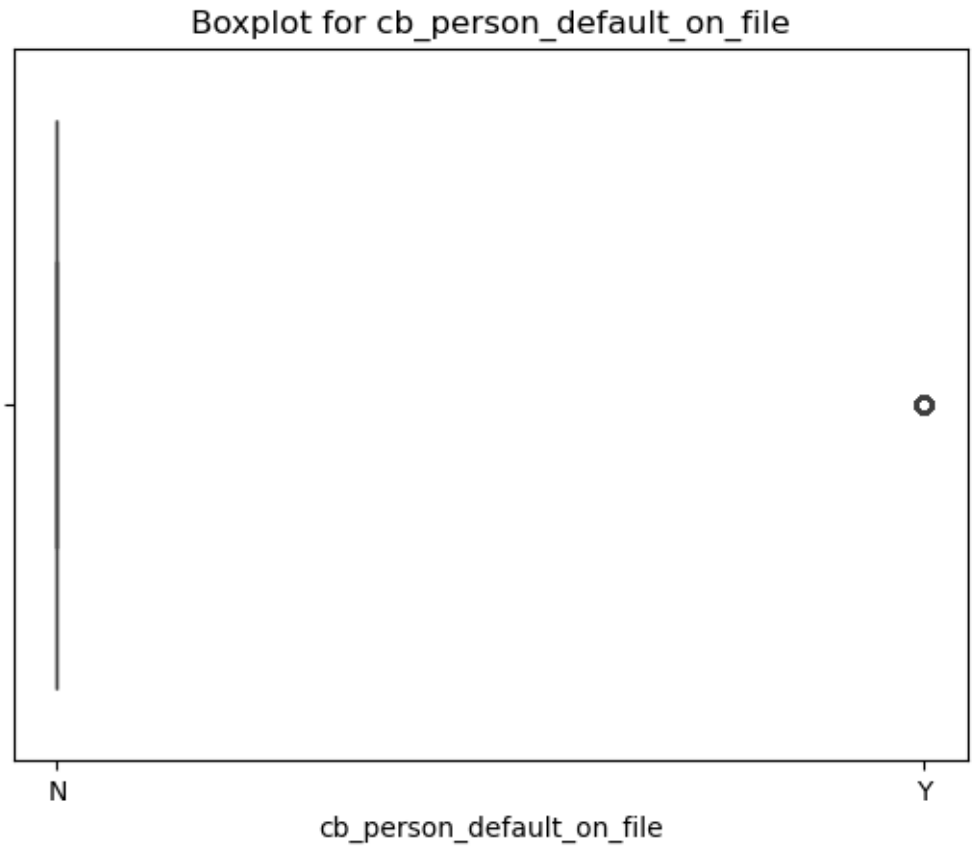

Boxplot for loan_intent
HOMEIMPROVEMENT
PERSONAL
EDUCATION
DEBTCONSOLIDATION
MEDICAL
VENTURE
loan_intent

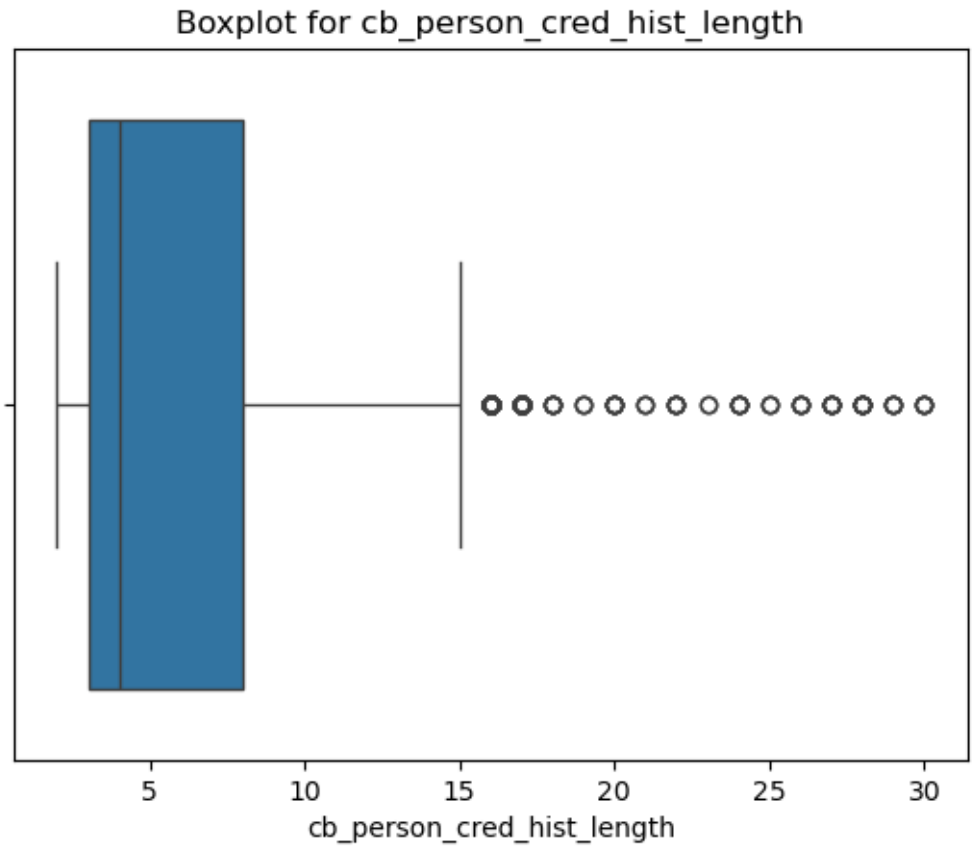

Boxplot for loan_grade
B
C
D
A
E
F
G
loan_grade


Boxplot for loan_amnt
0
5000
10000
15000
20000
25000
30000
35000
loan_amnt


Boxplot for loan_int_rate
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
loan_int_rate


Boxplot for loan_percent_income
0.0
0.2
0.4
0.6
0.8
loan_percent_income


Boxplot for cb_person_default_on_file
N
Y
cb_person_default_on_file



[972]:
```
new_data
```

[972]:

	person_age	person_income	person_home_ownership	person_emp_length	\
0	24	28000	OWN	6.0	
1	27	64000	RENT	0.0	
2	26	72000	MORTGAGE	10.0	
3	23	27996	RENT	7.0	
4	30	44500	RENT	2.0	
…	…	…	…	…	
8141	38	66000	RENT	1.0	
8142	27	39000	RENT	3.0	
8143	26	70000	RENT	5.0	
8144	24	10980	OWN	0.0	
8145	26	54600	MORTGAGE	10.0	

	loan_intent	loan_grade	loan_amnt	loan_int_rate	\
0	HOMEIMPROVEMENT	B	10000	10.37	
1	PERSONAL	C	10000	15.27	
2	EDUCATION	D	16000	11.03	
3	DEBTCONSOLIDATION	A	10000	11.03	

```
4               MEDICAL          E      13000          16.32
…                 …        …         …             …
8141           PERSONAL          D      10000          15.99
8142           PERSONAL          B      11000          11.03
8143           PERSONAL          C       4000          13.85
8144           PERSONAL          A       1500           7.29
8145           PERSONAL          B       4800          11.03

      loan_percent_income cb_person_default_on_file  \
0                    0.36                         N
1                    0.16                         Y
2                    0.22                         N
3                    0.36                         N
4                    0.29                         N
…                     …                          …
8141                 0.15                         Y
8142                 0.28                         N
8143                 0.06                         N
8144                 0.14                         N
8145                 0.09                         N

      cb_person_cred_hist_length
0                              2
1                             10
2                              3
3                              2
4                              6
…                            …
8141                          12
8142                           7
8143                           4
8144                           3
8145                           2

[8137 rows x 11 columns]
```

[973]:
```
new_data = new_data[new_data['person_age'] <= 80]
```

[974]:
```
new_data[new_data['person_income'] > 1_000_000]
```

[974]:
```
      person_age  person_income person_home_ownership  person_emp_length  \
1783          40        1200000              MORTGAGE                1.0
5863          42        2039784                  RENT                0.0
6107          47        1362000              MORTGAGE                9.0

     loan_intent loan_grade  loan_amnt  loan_int_rate  loan_percent_income  \
1783     MEDICAL          A      10000          11.03                 0.01
```

```
5863     VENTURE          C       8450          12.29                  0.00
6107     VENTURE          A       6600           7.74                  0.00

     cb_person_default_on_file  cb_person_cred_hist_length
1783                         N                          16
5863                         Y                          15
6107                         N                          17
```

[975]:
```
new_data['is_high_income'] = (new_data['person_income'] > 1_000_000).astype(int)
```

```
/var/folders/g_/dlksrxdd3pz88bqsmz91cx540000gn/T/ipykernel_28061/1733025878.py:1
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_data['is_high_income'] = (new_data['person_income'] >
1_000_000).astype(int)
```

[976]:
```
new_data[new_data['person_emp_length'] > 30]
```

[976]:
```
     person_age  person_income person_home_ownership  person_emp_length  \
990          21         192000              MORTGAGE              123.0

    loan_intent loan_grade  loan_amnt  loan_int_rate  loan_percent_income  \
990     VENTURE          A      20000           6.54                  0.1

    cb_person_default_on_file  cb_person_cred_hist_length  is_high_income
990                         N                           4               0
```

[977]:
```
median_emp_length_21 = new_data.loc[
    (new_data['person_age'] == 21) & (new_data['person_emp_length'] <= 100),
    'person_emp_length'
].median()

new_data.loc[
    (new_data['person_age'] == 21) & (new_data['person_emp_length'] > 100),
    'person_emp_length'
] = median_emp_length_21
```

[978]:
```
new_data.loc[990]
```

[978]:
```
person_age                    21
person_income             192000
person_home_ownership     MORTGAGE
person_emp_length              4.0
loan_intent               VENTURE
```

```
loan_grade                          A
loan_amnt                       20000
loan_int_rate                    6.54
loan_percent_income               0.1
cb_person_default_on_file           N
cb_person_cred_hist_length          4
is_high_income                      0
Name: 990, dtype: object
```

[979]:
```
new_data[
    (new_data['cb_person_cred_hist_length'] > 15) &
    (new_data['person_age'] < 30)
]
```

[979]:
```
Empty DataFrame
Columns: [person_age, person_income, person_home_ownership, person_emp_length,
loan_intent, loan_grade, loan_amnt, loan_int_rate, loan_percent_income,
cb_person_default_on_file, cb_person_cred_hist_length, is_high_income]
Index: []
```

[980]:
```
new_data['loan_amnt'].describe()
```

[980]:
```
count     8136.000000
mean      9593.390487
std       6377.942289
min        700.000000
25%       5000.000000
50%       8000.000000
75%      12000.000000
max      35000.000000
Name: loan_amnt, dtype: float64
```

[981]:
```
new_data['loan_int_rate'].describe()
```

[981]:
```
count    8136.000000
mean       11.051893
std         3.112534
min         5.420000
25%         8.490000
50%        11.030000
75%        13.160000
max        23.220000
Name: loan_int_rate, dtype: float64
```

[982]:
```
new_data[new_data['loan_percent_income'] > 0.5]
```

[982]:

	person_age	person_income	person_home_ownership	person_emp_length	\
59	28	28800	RENT	1.0	
284	54	53700	RENT	1.0	
621	25	24000	RENT	3.0	
665	27	30000	RENT	12.0	
732	24	43300	MORTGAGE	2.0	
…	…	…	…	…	
8000	27	47000	MORTGAGE	2.0	
8038	24	18000	MORTGAGE	0.0	
8047	23	47148	MORTGAGE	3.0	
8054	26	30000	RENT	5.0	
8134	25	25000	OWN	4.0	

	loan_intent	loan_grade	loan_amnt	loan_int_rate	\
59	EDUCATION	C	15000	14.65	
284	HOMEIMPROVEMENT	D	29850	11.03	
621	VENTURE	B	13250	10.37	
665	DEBTCONSOLIDATION	B	16000	10.99	
732	DEBTCONSOLIDATION	B	24000	10.62	
…	…	…	…	…	
8000	DEBTCONSOLIDATION	D	25000	14.96	
8038	DEBTCONSOLIDATION	B	10000	12.18	
8047	MEDICAL	A	25000	7.49	
8054	DEBTCONSOLIDATION	A	19075	7.49	
8134	MEDICAL	C	17500	14.22	

	loan_percent_income	cb_person_default_on_file	\
59	0.52	Y	
284	0.56	Y	
621	0.55	N	
665	0.53	N	
732	0.55	N	
…	…	…	
8000	0.53	Y	
8038	0.56	N	
8047	0.53	N	
8054	0.64	N	
8134	0.70	N	

	cb_person_cred_hist_length	is_high_income
59	8	0
284	28	0
621	2	0
665	6	0
732	2	0
…	…	…
8000	9	0

```
8038                        2               0
8047                        3               0
8054                        3               0
8134                        2               0

[76 rows x 12 columns]
```

[983]:
```
high_loan_threshold = new_data['loan_amnt'].quantile(0.75) # large loan
new_data['is_large_loan'] = (new_data['loan_amnt'] > high_loan_threshold).
 ↪astype(int)
```

```
/var/folders/g_/dlksrxdd3pz88bqsmz91cx540000gn/T/ipykernel_28061/2384363406.py:2
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_data['is_large_loan'] = (new_data['loan_amnt'] >
high_loan_threshold).astype(int)
```

[984]:
```
new_data_orig = new_data
```

Incorrect values have been handled, duplicates and missing entries have been removed — next, we proceed to encode and scale the variables.

[986]:
```
new_data = pd.get_dummies(new_data, columns=['person_home_ownership',␣
 ↪'loan_intent', 'loan_grade', 'cb_person_default_on_file'])
bool_columns = new_data.select_dtypes(include=['bool']).columns
new_data[bool_columns] = new_data[bool_columns].astype(int)
```

[987]:
```
scaler = RobustScaler()
new_data['person_age'] = scaler.fit_transform(new_data[['person_age']])
new_data['person_income'] = scaler.fit_transform(new_data[['person_income']])
new_data['loan_amnt'] = scaler.fit_transform(new_data[['loan_amnt']])
new_data['person_emp_length'] = scaler.
 ↪fit_transform(new_data[['person_emp_length']])
new_data['loan_int_rate'] = scaler.fit_transform(new_data[['loan_int_rate']])
new_data['cb_person_cred_hist_length'] = scaler.
 ↪fit_transform(new_data[['cb_person_cred_hist_length']])

scaler = MinMaxScaler()
new_data['person_age'] = scaler.fit_transform(new_data[['person_age']])
new_data['person_income'] = scaler.fit_transform(new_data[['person_income']])
new_data['loan_amnt'] = scaler.fit_transform(new_data[['loan_amnt']])
new_data['person_emp_length'] = scaler.
 ↪fit_transform(new_data[['person_emp_length']])
new_data['loan_int_rate'] = scaler.fit_transform(new_data[['loan_int_rate']])
```

```
new_data['cb_person_cred_hist_length'] = scaler.
  ↪fit_transform(new_data[['cb_person_cred_hist_length']])
```

[988]:
```
new_data
```

[988]:
```
      person_age  person_income  person_emp_length  loan_amnt  loan_int_rate  \
0       0.066667       0.011692           0.200000   0.271137       0.278090
1       0.116667       0.029377           0.000000   0.271137       0.553371
2       0.100000       0.033307           0.333333   0.446064       0.315169
3       0.050000       0.011690           0.233333   0.271137       0.315169
4       0.166667       0.019798           0.066667   0.358601       0.612360
…            …              …                  …          …              …
8141    0.300000       0.030360           0.033333   0.271137       0.593820
8142    0.116667       0.017096           0.100000   0.300292       0.315169
8143    0.100000       0.032325           0.166667   0.096210       0.473596
8144    0.066667       0.003331           0.000000   0.023324       0.105056
8145    0.100000       0.024759           0.333333   0.119534       0.315169

      loan_percent_income  cb_person_cred_hist_length  is_high_income  \
0                    0.36                    0.000000               0
1                    0.16                    0.285714               0
2                    0.22                    0.035714               0
3                    0.36                    0.000000               0
4                    0.29                    0.142857               0
…                       …                           …               …
8141                 0.15                    0.357143               0
8142                 0.28                    0.178571               0
8143                 0.06                    0.071429               0
8144                 0.14                    0.035714               0
8145                 0.09                    0.000000               0

      is_large_loan  person_home_ownership_MORTGAGE  …  loan_intent_VENTURE  \
0                 0                               0  …                    0
1                 0                               0  …                    0
2                 1                               1  …                    0
3                 0                               0  …                    0
4                 1                               0  …                    0
…               …                               …  …                    …
8141              0                               0  …                    0
8142              0                               0  …                    0
8143              0                               0  …                    0
8144              0                               0  …                    0
8145              0                               1  …                    0

      loan_grade_A  loan_grade_B  loan_grade_C  loan_grade_D  loan_grade_E  \
0                0             1             0             0             0
1                0             0             1             0             0
```

```
2                0              0              0              1              0
3                1              0              0              0              0
4                0              0              0              0              1
…              …              …              …              …              …
8141             0              0              0              1              0
8142             0              1              0              0              0
8143             0              0              1              0              0
8144             1              0              0              0              0
8145             0              1              0              0              0

      loan_grade_F  loan_grade_G  cb_person_default_on_file_N  \
0                0             0                            1
1                0             0                            0
2                0             0                            1
3                0             0                            1
4                0             0                            1
…              …             …                            …
8141             0             0                            0
8142             0             0                            1
8143             0             0                            1
8144             0             0                            1
8145             0             0                            1

      cb_person_default_on_file_Y
0                               0
1                               1
2                               0
3                               0
4                               0
…                             …
8141                            1
8142                            0
8143                            0
8144                            0
8145                            0

[8136 rows x 28 columns]
```

[989]:
```
new_pred_proba = model.predict_proba(new_data)[:, 1]
new_pred_class = (new_pred_proba >= optimal_threshold).astype(int)
```

[990]:
```
new_data_orig['default_probability'] = new_pred_proba
new_data_orig['loan_status'] = new_pred_class
```

```
/var/folders/g_/dlksrxdd3pz88bqsmz91cx540000gn/T/ipykernel_28061/2525261645.py:1
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_data_orig['default_probability'] = new_pred_proba
/var/folders/g_/dlksrxdd3pz88bqsmz91cx540000gn/T/ipykernel_28061/2525261645.py:2
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_data_orig['loan_status'] = new_pred_class
```

[991]:
```
new_data_orig
```

[991]:

	person_age	person_income	person_home_ownership	person_emp_length	\
0	24	28000	OWN	6.0	
1	27	64000	RENT	0.0	
2	26	72000	MORTGAGE	10.0	
3	23	27996	RENT	7.0	
4	30	44500	RENT	2.0	
…	…	…	…	…	
8141	38	66000	RENT	1.0	
8142	27	39000	RENT	3.0	
8143	26	70000	RENT	5.0	
8144	24	10980	OWN	0.0	
8145	26	54600	MORTGAGE	10.0	

	loan_intent	loan_grade	loan_amnt	loan_int_rate	\
0	HOMEIMPROVEMENT	B	10000	10.37	
1	PERSONAL	C	10000	15.27	
2	EDUCATION	D	16000	11.03	
3	DEBTCONSOLIDATION	A	10000	11.03	
4	MEDICAL	E	13000	16.32	
…	…	…	…	…	
8141	PERSONAL	D	10000	15.99	
8142	PERSONAL	B	11000	11.03	
8143	PERSONAL	C	4000	13.85	
8144	PERSONAL	A	1500	7.29	
8145	PERSONAL	B	4800	11.03	

	loan_percent_income	cb_person_default_on_file	\
0	0.36	N	
1	0.16	Y	
2	0.22	N	
3	0.36	N	
4	0.29	N	

```
…                    …                          …
8141                  0.15                       Y
8142                  0.28                       N
8143                  0.06                       N
8144                  0.14                       N
8145                  0.09                       N

      cb_person_cred_hist_length  is_high_income  is_large_loan  \
0                              2               0              0
1                             10               0              0
2                              3               0              1
3                              2               0              0
4                              6               0              1
…                            …               …              …
8141                          12               0              0
8142                           7               0              0
8143                           4               0              0
8144                           3               0              0
8145                           2               0              0

      default_probability  loan_status
0                0.068756            0
1                0.121867            0
2                0.173404            0
3                0.965158            1
4                0.727183            1
…                     …          …
8141             0.936748            1
8142             0.063403            0
8143             0.059898            0
8144             0.281420            0
8145             0.193761            0

[8136 rows x 15 columns]
```

[992]:
```
new_data_orig.to_csv('new_data_clients_predicted.csv', index=False)
```

[ ]:
```

```